Postgres Enterprise Manager

リリース *10.5.0*

EnterpriseDB Corporation

2026 年 07 月 03 日

目 次

第 1 章 Postgres compatibility **3**

第 2 章 Postgres Enterprise Manager release notes **5**

2.1 Postgres Enterprise Manager 10.5.0 release notes . . . 5
2.1.1 ハイライト . . . 5
2.1.2 機能強化 . . . 6
2.1.3 セキュリティ修正 . . . 8
2.1.4 の変更 . . . 11
2.1.5 バグ修正 . . . 12
2.2 Postgres Enterprise Manager 10.4.2 release notes . . . 16
2.2.1 セキュリティ修正 . . . 16
2.2.2 バグ修正 . . . 17
2.3 Postgres Enterprise Manager 10.4.1 release notes . . . 17
2.3.1 セキュリティ修正 . . . 18
2.3.2 バグ修正 . . . 18
2.4 Postgres Enterprise Manager 10.4.0 release notes . . . 18
2.4.1 ハイライト . . . 18
2.4.2 機能強化 . . . 18
2.4.3 の変更 . . . 20
2.4.4 バグ修正 . . . 20
2.5 Postgres Enterprise Manager 10.3.1 release notes . . . 21
2.5.1 の変更 . . . 22
2.5.2 バグ修正 . . . 22
2.6 Postgres Enterprise Manager 10.3.0 release notes . . . 23
2.6.1 ハイライト . . . 23
2.6.2 機能強化 . . . 23
2.6.3 の変更 . . . 25
2.6.4 バグ修正 . . . 26
2.7 Postgres Enterprise Manager 10.2.0 release notes . . . 29
2.7.1 ハイライト . . . 29
2.7.2 機能強化 . . . 29
2.7.3 の変更 . . . 30
2.7.4 バグ修正 . . . 31
2.8 Postgres Enterprise Manager 10.1.2 release notes . . . 32
2.9 Postgres Enterprise Manager 10.1.1 release notes . . . 33
2.9.1 バグ修正 . . . 33
2.10 Postgres Enterprise Manager 10.1.0 release notes . . . 34
2.10.1 ハイライト . . . 34
2.10.2 機能強化 . . . 34
2.10.3 の変更 . . . 35

2.10.4 バグ修正 . . . 36
2.11 Postgres Enterprise Manager 10.0.0 release notes . . . 37
2.11.1 ハイライト . . . 37
2.11.2 機能強化 . . . 38
2.11.3 の変更 . . . 39
2.11.4 バグ修正 . . . 40

第 3 章 Platform compatibility 41

第 4 章 Hardware requirements for installing the PEM server on Linux 43

第 5 章 PLANNING 45
5.1 Understanding Postgres Enterprise Manager components and architecture . . . 45
5.1.1 PEM アーキテクチャ . . . 46
5.1.1.1 PEM サーバー . . . 46
5.1.1.2 PEM エージェント . . . 48
5.1.1.3 PEM Web アプリケーション . . . 49
5.1.1.4 SQL プロファイラー拡張機能 . . . 49
5.2 Deployment considerations . . . 49
5.2.1 How PEM licensing works . . . 50
5.2.1.1 ライセンスルール . . . 50
5.2.1.2 例 . . . 50
5.2.2 PEM server and agent connection management mechanism . . . 51
5.2.2.1 PgBouncer なし . . . 51
5.2.2.2 PgBouncer を使用する場合 . . . 51
5.2.3 Authentication options . . . 53
5.2.3.1 1. 二要素認証 2FA . . . 53
5.2.3.2 2.Kerberos 認証 . . . 53
5.2.4 Deciding on an SSL model . . . 54
5.2.4.1 証明書管理モデル . . . 54
5.2.5 Securing your deployment . . . 55
5.2.5.1 Web server security configuration . . . 55
5.2.5.2 PEM application security configurations . . . 60
5.2.6 Web server installation options . . . 63
5.2.6.1 別のホスト上の PEM サーバーと Web サーバー . . . 63
5.2.6.2 同じホスト上の PEM サーバーと Web サーバー . . . 64
5.2.7 Sizing PEM deployments . . . 64
5.2.7.1 リソース要件に影響を与える要因 . . . 64
5.2.7.2 PEM T シャツのサイズ . . . 66
5.2.8 High availability patterns for PEM deployment . . . 67
5.2.8.1 PEM バックエンドデータベースの複数のインスタンスの実行 . . . 67
5.2.8.2 複数の PEM Web アプリケーションインスタンスの実行 . . . 68
5.2.8.3 HA PEM 展開のアップグレード . . . 68
5.2.8.4 ロードバランサー、プロキシ、および VIP . . . 68
5.2.8.5 リファレンスアーキテクチャ . . . 69

第 6 章 INSTALLING AND UPGRADING 75
6.1 Installing Postgres Enterprise Manager server . . . 75

6.1.1 Linux x86-64 (amd64) . . . 75
6.1.1.1 Red Hat Enterprise Linux RHEL とその派生 . . . 75
6.1.1.2 SUSE Linux Enterprise SLES . . . 75
6.1.1.3 Debian とデリバティブ . . . 75
6.1.2 Linux Installing Postgres Enterprise Manager server on Linux IBM Power (ppc64le) . . 75
6.1.2.1 Red Hat Enterprise Linux RHEL とその派生 . . . 75
6.1.2.2 SUSE Linux Enterprise SLES . . . 76
6.1.3 Linux Installing Postgres Enterprise Manager server on Linux AArch64 (ARM64) . . . 76
6.1.3.1 Red Hat Enterprise Linux RHEL とその派生 . . . 76
6.1.3.2 Debian とデリバティブ . . . 76
6.1.4 Windows . . . 76
6.1.4.1 Dependencies of the PEM Server and Agent on Linux . . . 76
6.1.4.2 Installing Postgres Enterprise Manager server on Linux x86 (amd64) . . . 80
6.1.4.3 Installing Postgres Enterprise Manager server on Linux AArch64 (ARM64) . . 101
6.1.4.4 Installing Postgres Enterprise Manager server on Linux IBM Power (ppc64le) . 105
6.1.4.5 Installing Postgres Enterprise Manager server on Windows . . . 111
6.1.4.6 Creating an EDB repository on an isolated network . . . 122
6.1.4.7 Configuring the PEM server on Linux . . . 123
6.1.4.8 Installing Postgres Enterprise Manager in HA Patterns . . . 129
6.2 Installing Postgres Enterprise Manager agent . . . 172
6.2.1 Linux x86-64 (amd64) . . . 172
6.2.1.1 Red Hat Enterprise Linux RHEL とその派生 . . . 172
6.2.1.2 SUSE Linux Enterprise SLES . . . 173
6.2.1.3 Debian とデリバティブ . . . 173
6.2.2 Linux Installing Postgres Enterprise Manager server on Linux IBM Power (ppc64le) . . 173
6.2.2.1 Red Hat Enterprise Linux RHEL とその派生 . . . 173
6.2.2.2 SUSE Linux Enterprise SLES . . . 173
6.2.3 Linux Installing Postgres Enterprise Manager server on Linux AArch64 (ARM64) . . . 173
6.2.3.1 Red Hat Enterprise Linux RHEL とその派生 . . . 173
6.2.3.2 Debian とデリバティブ . . . 173
6.2.4 Windows . . . 173
6.2.4.1 Installing Postgres Enterprise Manager agent on Linux x86 (amd64) . . . 174
6.2.4.2 Installing Postgres Enterprise Manager agent on Linux AArch64 (ARM64) . . . 178
6.2.4.3 Installing Postgres Enterprise Manager agent on Linux IBM Power (ppc64le) . . 179
6.2.4.4 Installing Postgres Enterprise Manager agent on Windows . . . 181
6.3 Configuring user authentication in PEM . . . 184
6.3.1 Configuring the PEM server to use two-factor authentication . . . 184
6.3.1.1 メールサーバーの構成 . . . 185
6.3.2 Configuring the PEM server to use Kerberos authentication . . . 185
6.3.2.1 1.Kerberos、PEM サーバー、および PEM バックエンドデータベースをインストールします . . . 185
6.3.2.2 2.Kerberos サーバーにプリンシパルを追加する . . . 186
6.3.2.3 3.Kerberos サーバーからキーテーブルを抽出します . . . 186
6.3.2.4 4.PEM バックエンドデータベースサーバーの構成 . . . 187
6.3.2.5 5. 最初のチケットを取得して表示します . . . 188
6.3.2.6 6.PEM サーバーの構成 . . . 188
6.3.2.7 7. ブラウザ設定 . . . 189

6.3.3 Configuring the PEM server to use Windows Active Directory domain services for Kerberos authentication (SSPI) . . . 190
6.3.3.1 1. Active Directory、PEM サーバー、および PEM バックエンドデータベースサーバーのインストール . . . 190
6.3.3.2 2. サービスプリンシパルとマップする Active Directory にユーザーを作成する . . . 191
6.3.3.3 3. Active Directory からキーテーブルを抽出する . . . 193
6.3.3.4 4.PEM バックエンドデータベースサーバーの構成 . . . 195
6.3.3.5 5. 最初のチケットを取得して表示します . . . 196
6.3.3.6 6.PEM サーバーの構成 . . . 196
6.4 Connection pooling using PgBouncer . . . 198
6.4.1 Preparing the PEM database server . . . 199
6.4.1.1 前提条件 . . . 199
6.4.1.2 PgBouncer-PEM 接続のユーザーとロールの作成 . . . 199
6.4.1.3 構成ファイルを更新して PgBouncer-PEM 接続を許可する . . . 200
6.4.1.4 PgBouncer-PEM 認証の SSL キーと証明書の作成 . . . 201
6.4.2 Configuring PgBouncer . . . 202
6.4.2.1 前提条件 . . . 202
6.4.2.2 PgBouncer の構成 . . . 203
6.4.3 Configuring the PEM agent . . . 205
6.4.3.1 前提条件 . . . 205
6.4.3.2 新しい PEM エージェントの構成 . . . 205
6.4.3.3 既存の PEM エージェントの構成 . . . 206
6.5 Upgrade and migration . . . 207
6.5.1 Upgrading a PEM installation . . . 207
6.5.1.1 Upgrading a PEM installation on a Linux host . . . 207
6.5.1.2 Upgrading a PEM installation on a Windows host . . . 210
6.5.1.3 Upgrading an HA PEM installation . . . 212
6.5.2 PEM upgrade best practices and remediation . . . 214
6.5.2.1 PEM アップグレードで権限を保持する方法 . . . 214
6.5.2.2 してはいけないこと . . . 215
6.5.2.3 アップグレード前の準備 . . . 216
6.5.2.4 アップグレード後の検証 . . . 218
6.5.2.5 すでに影響を受けているクラスターの修復 . . . 219
6.5.2.6 例 所有権を変更したアップグレード . . . 220
6.5.3 Upgrading the PEM backend Postgres database . . . 221
6.5.4 Adding entries for PEM Agens and admins to connect to PEM server . . . 225
6.5.5 Adding entries (localhost) for PEM Agens and admins to connect to PEM server . . . 225
6.5.6 Upgrading SQL Profiler . . . 226
6.5.7 Upgrading the PEM web server . . . 226
6.5.7.1 Linux での Web サーバーのアップグレード . . . 226
6.5.7.2 Windows での PEM-HTTPD のアップグレード . . . 227
6.5.8 Moving the PEM server . . . 228
6.5.8.1 PEM エージェントが Linux にある場合 . . . 234
6.5.8.2 PEM エージェントが Windows にある場合 . . . 234
6.6 Splitting an existing PEM server . . . 235
6.6.1 新しいホストに PEM をインストールする . . . 236

6.6.2 既存の PEM インスタンスからすべてのデータをコピーします . . . 236
6.6.3 新しい PEM インスタンスの pg_hba.conf ファイルを更新します . . . 236
6.6.4 新しいサーバーで SSL を構成し、PEM エージェントを再構成します . . . 236
6.6.4.1 アプローチ 2 –異なるサーバー証明書を使用するが、同じ CA を使用する . 237
6.6.4.2 アプローチ 3 –すべてのサーバーとエージェントの証明書を置き換えます . 238
6.6.4.3 アプローチ 4 –外部 CA からの証明書のみを使用する . . . 238
6.6.5 各 PEM インスタンスで切断されたエージェントを無効にする . . . 238
6.6.6 PEM インスタンスを検証する . . . 239
6.7 Uninstalling . . . 239
6.7.1 Uninstalling Postgres Enterprise Manager components on Linux . . . 239
6.7.1.1 RHEL、Rocky Linux、または AlmaLinux ホストからの PEM コンポーネントのアンインストール . . . 240
6.7.1.2 Debian または Ubuntu ホストからの PEM コンポーネントのアンインストール 240
6.7.1.3 SLES ホストからの PEM コンポーネントのアンインストール . . . 240
6.7.2 Uninstalling Postgres Enterprise Manager components on Windows . . . 241
6.8 Troubleshooting . . . 241
6.8.1 既知のインストールと構成の問題 . . . 241
6.8.1.1 既存の mod_wsgi との競合 . . . 241
6.8.1.2 JIT からのエラー . . . 242
6.8.1.3 RHEL Python 3.9 標準ライブラリのバグによるクラッシュ . . . 242
6.8.1.4 Apache HTTPD での TRACK / TRACE メソッドのブロック . . . 243
6.8.2 PEM サーバーの再構成 . . . 243
6.8.3 PEM Web アプリケーションがロードされません . . . 244
6.8.4 PostgreSQL サーバーへの接続エラー . . . 244
6.8.5 PEM Web サーバーの接続性チェック . . . 245
6.8.6 SQL エラー-リレーション「server」の列「ssl」が存在しません . . . 245
6.8.7 SSL エラー-不完全な sslutils のアップグレード . . . 246
6.8.8 Nginx SSL_read() 失敗エラー . . . 246

第 7 章 ADMINISTERING 249
7.1 Changing the default port . . . 249
7.1.1 RHEL の場合 . . . 249
7.1.1.1 SELinux 構成を変更する . . . 249
7.1.1.2 Web サーバー NGINX の構成 . . . 249
7.1.1.3 Web サーバー Apache HTTPD の構成 . . . 250
7.1.2 Debian および Ubuntu の場合 . . . 250
7.1.2.1 Web サーバー NGINX の構成 . . . 250
7.1.2.2 Web サーバー Apache HTTPD の構成 . . . 251
7.2 Registering a PEM agent . . . 252
7.2.1 PEM エージェントを登録する方法 . . . 252
7.2.2 pemworker ユーティリティを使用した PEM エージェントの登録 . . . 252
7.2.2.1 Linux . . . 252
7.2.2.2 Windows . . . 252
7.2.2.3 pemworker ユーティリティの認証 . . . 254
7.2.2.4 PEM エージェントサービスを有効にする . . . 255
7.2.2.5 Linux . . . 255
7.2.2.6 Windows . . . 255

7.2.3 PEM エージェント登録の検証 . . . 255
7.2.3.1 コマンド構文 . . . 256
7.2.3.2 終了コードの理解 . . . 256
7.2.3.3 使用例 . . . 257
7.2.4 PEM エージェントの登録解除 . . . 257
7.2.5 高度な使用法 . . . 258
7.2.5.1 接続文字列を使用する . . . 258
7.2.5.2 登録接続文字列 . . . 258
7.2.5.3 エージェント接続文字列 . . . 259
7.2.5.4 エージェント ID の設定 . . . 259
7.2.5.5 デフォルト構成のオーバーライド-例 . . . 259
7.2.5.6 エージェント ID の設定 . . . 260
7.2.5.7 既存の SSL 証明書とキーの提供 . . . 260
7.2.5.8 非 root ユーザーアカウントを使用して Linux で PEM エージェントを登録する . . . 261
7.3 Registering a Postgres server . . . 263
7.3.1 Postgres サーバーを登録するとどうなりますか? . . . 263
7.3.2 pemworker ユーティリティを使用してサーバーを登録する . . . 263
7.3.2.1 pemworker 接続の追加パラメーターの指定 . . . 266
7.3.2.2 例 . . . 267
7.3.2.3 pemworker ユーティリティを使用してサーバーの登録を解除する . . . 267
7.3.3 自動サーバー検出へのサーバーの登録 . . . 268
7.3.4 データベースサーバーの手動登録 . . . 269
7.3.5 接続とバインディングの検証 . . . 276
7.4 Managing a PEM server . . . 276
7.4.1 PEM サーバーとエージェントの起動と停止 . . . 277
7.4.2 Linux での PEM サーバーまたは PEM エージェントの制御 . . . 277
7.4.3 Windows での PEM サーバーまたは PEM エージェントの制御 . . . 278
7.4.4 Linux での Web サーバーの制御 . . . 278
7.4.4.1 NGINX . . . 278
7.4.4.2 Apache HTTPD . . . 279
7.4.5 Windows での Web サーバーの制御 . . . 279
7.4.6 PEM 認証の管理 . . . 279
7.4.7 PEM サーバー構成の編集 . . . 280
7.4.8 セキュリティの管理 . . . 280
7.4.8.1 ログインロール . . . 280
7.4.8.2 グループロール . . . 281
7.4.8.3 PEM の事前定義されたロールを使用して PEM 機能へのアクセスを管理する 281
7.4.8.4 REST API アクセスに必要なロール . . . 284
7.4.8.5 チームロールを使用する . . . 285
7.4.8.6 オブジェクト権限 . . . 285
7.4.9 サーバー構成 . . . 286
7.4.10 サーバー構成パラメーター - リファレンス . . . 286
7.5 Managing Postgres servers . . . 296
7.5.1 監視対象データベースサーバーの起動と停止 . . . 296
7.5.2 pg_hba.conf ファイルの変更 . . . 296
7.5.3 データベースとオブジェクトの作成と保守 . . . 298

7.5.4 テンプレート Linux サービススクリプト . . . 298
7.6 Managing a PEM agent . . . 299
7.6.1 Managing job notifications . . . 299
7.6.1.1 ジョブレベルでジョブ通知を設定する . . . 299
7.6.1.2 エージェントレベルでのジョブ通知の設定 . . . 299
7.6.1.3 サーバーレベルでジョブ通知を設定する . . . 300
7.6.2 Managing scheduled jobs . . . 300
7.6.2.1 ジョブに関する一般的な情報を提供します . . . 300
7.6.2.2 ジョブステップを定義する . . . 301
7.6.2.3 ジョブスケジュールを定義する . . . 302
7.6.2.4 メール通知を設定する . . . 302
7.6.3 Modifying agent configuration . . . 303
7.6.3.1 構成ファイルの内容 . . . 303
7.6.3.2 レジストリの内容 . . . 307
7.6.3.3 アラートスレッドについて . . . 309
7.6.4 Reviewing or modifying agent properties . . . 310
7.6.5 Running multiple PEM agent services on a single server . . . 310
7.6.5.1 前提条件 . . . 311
7.6.5.2 Linux . . . 311
7.6.5.3 Windows . . . 312
7.6.5.4 エージェント登録の確認 . . . 312
7.6.6 Setting agent privileges . . . 313
7.6.6.1 root ユーザーと非 root ユーザー . . . 313
7.6.6.2 オペレーティングシステムの権限の制限によって影響を受ける機能 . . . 314
7.6.6.3 データベース特権の制限によって影響を受ける機能 . . . 315
7.6.6.4 エラー処理 . . . 316
7.7 Scheduled system jobs . . . 317
7.7.1 システムジョブリファレンス . . . 317
7.8 Managing SSL certificates . . . 318
7.8.1 Web サーバー証明書 . . . 318
7.8.1.1 Web サーバー証明書 NGINX の置き換え . . . 318
7.8.1.2 Web サーバー証明書の置き換え Apache HTTPD . . . 319
7.8.2 PEM バックエンドデータベースサーバーとエージェントの接続証明書 . . . 319
7.8.2.1 PEM サーバー上の証明書とキーファイル . . . 319
7.8.2.2 PEM エージェントの証明書とキーファイル . . . 320
7.8.2.3 PEM 証明書の更新 . . . 320
7.8.3 共通エージェント証明書とキーペアを生成する . . . 321
7.8.4 信頼できる CA によって署名された証明書とキーを使用する . . . 321
7.8.4.1 サーバー SSL 証明書を、信頼できる CA によって署名された証明書に置き換えます。 . . . 321
7.8.4.2 エージェント SSL 証明書を、信頼できる CA によって署名された証明書に置き換えます。 . . . 323
7.8.5 証明書のテスト . . . 324
7.8.5.1 ルート証明書と照合して証明書を検証する . . . 324
7.8.5.2 PEM バックエンドデータベースへのテスト接続を作成する . . . 324
7.9 Managing configuration settings . . . 331
7.10 Troubleshooting agent issues . . . 331

7.10.1 削除された PEM エージェントの復元 . . . 331
7.10.2 構成ファイルパスの更新エージェントのステータスが不明と表示されます . . . 332
7.10.3 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 . . . 332

第 8 章 USING **335**

8.1 Accessing the web interface . . . 335
8.1.1 PEM クライアントオブジェクトブラウザ . . . 337
8.1.2 PEM タブ付きブラウザウィンドウ . . . 338
8.1.3 チャート、グラフ、テーブルコントロールを使用する . . . 341
8.1.3.1 グラフィックのパーソナライズ . . . 341
8.1.4 ブラウザツールバー . . . 342
8.1.5 PEM メニューバー . . . 342
8.1.5.1 ファイルメニュー . . . 342
8.1.5.2 オブジェクトメニュー . . . 343
8.1.5.3 管理メニュー . . . 344
8.1.5.4 ダッシュボードメニュー . . . 344
8.1.5.5 [ツール] メニュー . . . 345
8.1.5.6 ヘルプメニュー . . . 346
8.1.6 チャート、グラフ、テーブルの制御とカスタマイズ . . . 346
8.2 Connecting and disconnecting a database server . . . 347
8.2.1 サーバーからの切断 . . . 347
8.3 Monitoring performance . . . 347
8.3.1 Probes . . . 350
8.3.1.1 システムプローブ . . . 350
8.3.1.2 プローブのカスタマイズ . . . 350
8.3.2 Alerts . . . 368
8.3.2.1 グローバルダッシュボードを介したアラートの表示 . . . 369
8.3.2.2 アラートダッシュボードを介したアラートの表示 . . . 369
8.3.2.3 アラートダッシュボードのカスタマイズ . . . 370
8.3.3 Managing monitoring settings using profiles . . . 398
8.3.3.1 プロファイルを使用する目的 . . . 399
8.3.3.2 主要な概念 . . . 399
8.3.3.3 管理プロファイルへのアクセス . . . 400
8.3.3.4 プロファイルの割り当てページ . . . 405
8.3.4 Notifications . . . 407
8.3.4.1 SMTP . . . 407
8.3.4.2 Webhooks . . . 410
8.3.4.3 SNMP . . . 416
8.3.4.4 Nagios . . . 417
8.3.5 Capacity Manager . . . 419
8.3.5.1 Capacity Manager テンプレート . . . 421
8.3.5.2 Capacity Manager メトリック - リファレンス . . . 422
8.3.6 Audit Manager . . . 425
8.3.6.1 EDB Postgres Advanced Server インスタンスサービス ID の設定 . . . 426
8.3.6.2 EDB Audit Configuration プローブの設定 . . . 426
8.3.6.3 Audit Manager を使用した監査ログの構成 . . . 427

8.3.6.4 Audit Log ダッシュボードでのログの表示 . . . 429
8.3.7 Log Manager . . . 430
8.3.7.1 Server Log Analysis ダッシュボードの確認 . . . 434
8.3.8 Charts . . . 435
8.3.8.1 カスタムチャートの作成 . . . 436
8.3.8.2 Capacity Manager テンプレートのインポート . . . 438
8.3.9 Dashboards . . . 439
8.3.9.1 ダッシュボードの概要 . . . 439
8.3.9.2 使用可能なダッシュボード . . . 440
8.3.9.3 ダッシュボード構成 . . . 442
8.3.9.4 カスタムダッシュボード . . . 443
8.3.10 Remote monitoring . . . 443
8.4 Monitoring Barman . . . 444
8.4.1 Barman をモニタリングするための前提条件 . . . 444
8.4.2 Barman サーバーの構成 . . . 445
8.4.2.1 PEM Web クライアントを使用する . . . 445
8.4.2.2 構成 . . . 445
8.4.2.3 編集 . . . 445
8.4.2.4 pemworker コマンドラインを使用する . . . 446
8.4.3 PEM ダッシュボードでの Barman サーバーの詳細の表示 . . . 447
8.5 Monitoring EDB Postgres Distributed . . . 447
8.6 Cluster topology . . . 448
8.6.1 トポロジ検出の仕組み . . . 449
8.6.1.1 必須プローブ . . . 449
8.6.1.2 クラスタータイプの検出 . . . 449
8.6.2 トポロジービューへのアクセス . . . 450
8.6.2.1 必要な権限 . . . 450
8.6.3 トポロジ表示を理解する . . . 450
8.6.3.1 クラスターリスト左側のサイドバー . . . 450
8.6.3.2 サポートされているクラスタータイプ . . . 450
8.6.3.3 トポロジグラフメインパネル . . . 451
8.6.4 インタラクティブ SVG ビュー . . . 451
8.6.4.1 パンとズーム . . . 451
8.6.4.2 フルスクリーンモード . . . 451
8.6.4.3 SVG エクスポート . . . 451
8.6.4.4 ノードカードの詳細 . . . 451
8.6.4.5 レンダラー設定 . . . 452
8.6.5 PGD/ BDR クラスター . . . 452
8.6.5.1 ポリゴンとフォースグラフのレイアウト . . . 452
8.6.5.2 サブグループ . . . 453
8.6.6 未登録のレプリカでの作業 . . . 453
8.6.7 トポロジデータの更新 . . . 453
8.6.8 障害対応 . . . 453
8.6.9 ベストプラクティス . . . 454
8.6.10 トポロジデータの SQL クエリー . . . 455
8.6.10.1 検出されたすべてのクラスターをリストする . . . 455
8.6.10.2 ロールを持つクラスターメンバーを表示する . . . 455

8.6.10.3 レプリケーションラグの表示 PSR、EFM、および Patroni クラスター . . . 456
8.6.10.4 レプリケーションラグ PGD クラスターの表示 . . . 456
8.6.10.5 完全なトポロジの概要すべてのクラスタータイプ . . . 457
8.7 Monitoring Failover Manager . . . 458
8.7.1 プライマリノードの置き換え . . . 459
8.7.2 スイッチオーバー EFM クラスター . . . 459
8.8 Monitoring Replication Server . . . 460
8.9 Monitoring event history . . . 460
8.9.1 なぜ重要なのでしょうか？ . . . 461
8.9.2 コア特典 . . . 461
8.9.3 サポートされている機能 . . . 461
8.9.3.1 プロファイル管理 . . . 462
8.9.3.2 アラートブラックアウト管理 . . . 462
8.9.3.3 高可用性 HA モニタリング . . . 463
8.9.4 一般的な運用シナリオ . . . 463
8.9.5 イベントエントリーリファレンス . . . 463
8.9.5.1 1. プロファイルの割り当て . . . 463
8.9.5.2 2. プロファイルのクローン . . . 464
8.9.5.3 3. プロファイルのエクスポート . . . 464
8.9.5.4 4. プロファイルのインポート新しい名前 . . . 465
8.9.5.5 5. プロファイルインポート上書き → 下書き . . . 465
8.9.5.6 6. アラートブラックアウト有効チェックボックス . . . 465
8.9.5.7 7. アラートブラックアウトは無効チェックボックス . . . 466
8.9.5.8 8. スケジュールされた停電が作成されましたダイアログ . . . 466
8.9.5.9 9. 計画的な停電の削除ダイアログ . . . 467
8.9.5.10 10. 手動ブラックアウトの有効期限が切れました . . . 467
8.9.5.11 11. 計画的な停電が完了しました . . . 467
8.9.5.12 12. 自動ブラックアウトが有効になっています . . . 468
8.9.5.13 13. 自動ブラックアウトは無効です . . . 468
8.9.5.14 14.HA スイッチオーバー . . . 468
8.9.6 最近のアクティビティを一目で見る方法 . . . 469
8.9.6.1 最近のプロフィールイベント . . . 469
8.9.6.2 最近のプロファイルクローンイベント . . . 469
8.9.6.3 最近のアラートブラックアウトの変更 . . . 469
8.9.6.4 最近の HA スイッチオーバー . . . 470
8.9.7 詳細フィールドの読み取り . . . 470
8.9.8 考慮事項 . . . 471
8.10 Tuning performance . . . 471
8.10.1 Performance Diagnostic . . . 472
8.10.1.1 前提条件 . . . 472
8.10.1.2 パフォーマンス診断ダッシュボードを使用する . . . 473
8.11 Profiling workloads . . . 476
8.11.1 Installing the SQL Profiler extension . . . 476
8.11.1.1 パッケージのインストール . . . 476
8.11.1.2 拡張機能の有効化 . . . 477
8.11.2 Using SQL Profiler . . . 478
8.11.2.1 SQL プロファイラーの権限 . . . 478

8.11.2.2 トレースの作成 . . . 478
8.11.2.3 既存のトレースを開く . . . 479
8.11.2.4 トレースのフィルタリング . . . 479
8.11.2.5 トレースの削除 . . . 480
8.11.2.6 スケジュールされたトレースの表示 . . . 481
8.12 Reports . . . 481
8.12.1 Custom report templates . . . 481
8.12.2 Viewing system reports . . . 481
8.12.2.1 Custom report templates . . . 482
8.12.2.2 Viewing system reports . . . 488
8.13 PEM command line interface . . . 498
8.13.1 例 . . . 501
8.14 EDB Postgres Enterprise Manager REST APIs v1.0.0 . . . 501
8.14.1 トークン . . . 502
8.14.1.1 トークン生成 . . . 502
8.14.1.2 パラメーター . . . 503
8.14.1.3 応答 . . . 503
8.14.1.4 レスポンススキーマ . . . 504
8.14.1.5 レスポンスヘッダー . . . 505
8.14.1.6 delete_*token* . . . 505
8.14.1.7 パラメーター . . . 506
8.14.1.8 応答 . . . 506
8.14.1.9 レスポンススキーマ . . . 506

第 9 章 DEVELOPING **1295**
9.1 Using the Query tool 1295
9.1.1 クエリツールツールバー 1296
9.1.2 SQL エディターパネル 1298
9.1.3 [データ出力] パネル 1299
9.1.3.1 [データ出力] タブ 1299
9.1.3.2 「Explain」タブ 1300
9.1.3.3 グラフィカルタブ 1300
9.1.3.4 [分析] タブ 1300
9.1.3.5 [統計] タブ 1301
9.1.3.6 [メッセージ] タブ 1301
9.1.3.7 [通知] タブ 1301
9.1.4 Query History パネル 1301
9.1.5 接続ステータス 1303
9.1.6 マクロ 1303
9.2 Using the Schema Diff tool 1304
9.2.1 Schema Diff Object Comparison パネル 1304
9.2.2 Schema Diff DDL Comparison パネル 1305
9.3 Using the ERD tool 1306
9.3.1 ツールバー 1306
9.3.2 ファイルオプション 1306
9.3.3 エクスポートオプション 1307
9.3.4 編集オプション 1307

9.3.5 テーブルリレーションシップオプション 1307
9.3.6 ユーティリティオプション 1308
9.3.7 ズームオプション 1308
9.3.8 テーブルダイアログ 1308
9.3.9 テーブルノード 1308
9.3.10 1 対多のリンクダイアログボックス 1309
9.3.11 多対多のリンクダイアログボックス 1309
9.3.12 テーブルリンク 1310
9.3.13 テーブルノート 1310
9.4 PL debugger 1311
9.4.1 インコンテキストデバッグ 1311
9.4.2 ダイレクトデバッグ 1311
9.4.3 デバッガーを使用する 1313

Postgres Enterprise ManagerPEM 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、メンテナンス、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。

第1章　Postgres compatibility

PEM 10.x でサポートされている Postgres のバージョン

	Monitored Instance	Backend Instance
EDB Postgres Advanced Server (EPAS)	14, 15, 16, 17, 18	14, 15, 16, 17, 18
PostgreSQL (PG)	14, 15, 16, 17, 18	14, 15, 16, 17, 18
EDB Postgres Extended Server (PGE)	14, 15, 16, 17, 18	14, 15, 16, 17, 18

第2章　Postgres Enterprise Manager release notes

Postgres Enterprise Manager ドキュメントでは、Postgres Enterprise Manager 10 の最新バージョンについて説明しています。

Postgres Enterprise Manager version	Release Date
Postgres Enterprise Manager 10.5.0 release notes	22 Jun 2026
Postgres Enterprise Manager 10.4.2 release notes	14 May 2026
Postgres Enterprise Manager 10.4.1 release notes	09 Apr 2026
Postgres Enterprise Manager 10.4.0 release notes	20 Mar 2026
Postgres Enterprise Manager 10.3.1 release notes	25 Feb 2026
Postgres Enterprise Manager 10.3.0 release notes	25 Nov 2025
Postgres Enterprise Manager 10.2.0 release notes	14 Aug 2025
Postgres Enterprise Manager 10.1.2 release notes	21 Jul 2025
Postgres Enterprise Manager 10.1.1 release notes	07 Jul 2025
Postgres Enterprise Manager 10.1.0 release notes	15 May 2025
Postgres Enterprise Manager 10.0.0 release notes	24 Mar 2025

2.1 Postgres Enterprise Manager 10.5.0 release notes

リリース日 2026 年 6 月 22 日

!!!PEM のサイズとスケーラビリティーに注意してください PEM 10.5 には、

sizing guidance の変更をもたらしたパフォーマンスの強化が含まれています。これを確認して、システムが PEM 10.5 に適切なサイズであることを確認することをお勧めします。

!!!注 CLI オプション解析の変更 PEM 10.5 は、`pemworker` CLI および `configure_pem_server.sh` のより堅牢なオプション解析を備えています。以前に受け入れられたオプションの一部の組み合わせは受け入れられない場合があるため、アップグレード時にこれらの CLI を使用するスクリプトまたは自動化を確認してください。

Postgres Enterprise Manager 10.5.0 の新機能、機能強化、バグ修正、およびその他の変更には次のものが含まれます。

2.1.1 ハイライト

- プロファイルの割り当てと作成の UX の向上
- PEM サーバー自分自身に対してカスタムレポートを実行する機能を含む、新しいレポート機能の複数の機能強化

- 特に複数のアラートスレッドがあるシステムで、アラート評価パフォーマンスが大幅に向上しました

2.1.2 機能強化

Description	Addresses
計画的なダウンタイムと計画外のダウンタイム分類で可用性を追跡するサーバーダウンタイムレポートを導入しました。ハートビートギャップを監視することによりサーバーの可用性を追跡し、ブラックアウト期間を使用して計画的対計画外のダウンタイム分類を含むアップタイム/ダウンタイム分析を提供するサーバーダウンタイムレポートを導入しました。	
呼び出しごとに複数のオブジェクトのプローブデータと履歴 REST API エンドポイント v17 を追加しました。プローブデータと履歴 REST API エンドポイント v17 を追加しました。これは、単一の呼び出しで複数のオブジェクトの `pemdata `および `pemhistory`テーブルからの監視データを公開し、数値プローブ ID と内部名前アドレス指定の両方をサポートします。	
レプリケーションクラスターのインタラクティブ SVG 図を使用したクラスタートポロジの検出と視覚化が導入されました。クラスタートポロジの検出と可視化は、PostgreSQL レプリケーションクラスター物理ストリーミングレプリケーション、パトロニ、PGD、EFM を自動的に検出し、PEM ブラウザツリーにインタラクティブ SVG トポロジ図をレンダリングします。カスケードレプリケーション、ラグ表示、ロールバッジのサポート。	
プロファイルの割り当てをインラインまたは一括で表示および変更するためのプロファイルの割り当てページを追加しました。新しいプロファイルの割り当てページ 管理モニタリングプロファイルの割り当て を追加しました。このページは、サーバーグループとクラスターごとにグループ化されたツリービューにすべてのサーバーとエージェントを表示し、許可されたユーザーがインラインまたは複数選択を介して一括でプロファイルの割り当てを表示および変更できます。	
行レベルのセキュリティポリシーを使用するレポートテンプレートにチームベースの RBAC フィルタリングを実装しました。レポートテンプレートにチームベースの RBAC フィルタリングを実装し、管理者はレポートを PEM 専用として指定できますチームベースの可視性を適用する行レベルのセキュリティポリシーを使用して PEM データベースで実行されます。	
レポートテンプレートに繰り返しセクションのサポートを追加し、グループクエリ行を反復します。レポートテンプレートに繰り返しセクションのサポートを追加し、レポートセクションが行ごとのパラメーター置換を使用してグループクエリから行を反復して、単一のセクション内に構造化されたマルチインスタンス出力を生成できるようにします。	
PEM インストール全体で監視プロファイルを複製するためのプロファイルのインポート/エクスポートエンドポイントを追加しました。プロファイルのインポート/エクスポートエンドポイントを追加して、ユーザーが監視プロファイルを JSON ファイルとしてエクスポートし、他の PEM インストールにインポートして、環境間で簡単に構成をレプリケートできるようにします。	

次のページに続く

表 2 – 前のページからの続き

Description	Addresses
[Add Profile] ダイアログを介して既存のプロファイルのクローンを作成する機能を追加しました。 [Add Profile] ダイアログを介して既存のプロファイルのクローンを作成する機能が追加され、管理者は手動再構成なしで既存の構成に基づいて新しいプロファイルを作成できます。	
PEM REST API 書き込みエンドポイントの管理ロールを必要とするアプリケーションレイヤーの承認ゲートを追加しました。 PEM REST API 書き込みエンドポイントのアプリケーションレイヤーの承認ゲートを追加し、呼び出し元は書き込み操作に関連する PEM 管理ロールたとえば、資産管理またはアラート管理を保持する必要があります。	
REST API v17 で監視対象オブジェクトの名前による検索機能を導入しました。 API v17 の名前による検索機能が導入され、ユーザーは REST API を介して名前で監視対象オブジェクトエージェント、サーバー、データベースを検索できます。	
`server_id`および`agent_id`をシステム構成レポートの遅延アラートセクションに追加しました。遅延アラートセクションに`server_id`および`agent_id`を含めるようにシステム構成レポートを強化し、遅延したアラート処理の影響を受けるサーバーまたはエージェントを特定しやすくしました。	
マルチサーバーとマルチエージェントの選択をアラート履歴レポートモーダルに追加しました。アラート履歴レポートモーダルで複数のサーバーとエージェントを選択するためのサポートを追加し、ユーザーは 1 回の実行で複数の監視対象ターゲットにまたがるレポートを生成できます。	57611
レポート履歴テーブルと実行レポートヘッダーにデータベース名表示を追加しました。レポート履歴テーブルと実行レポートヘッダーにデータベース名表示を追加し、レポートがどのデータベースに対して実行されたかに関する明確なコンテキストをユーザーに提供します。	
`is_pem_report`フラグを追加して、コア使用状況レポートが PEM データベースで実行されるようにしました。`is_pem_report`フラグをコア使用状況レポートに追加し、PEM データベースで実行され、適切なレポートカテゴリに表示されます。	
アラートダッシュボードに「Show Acknowledged Alerts」設定を再導入しました。アラートダッシュボードに「Show Acknowledged Alerts」設定を再導入し、ユーザーがモニタリングビューで確認されたアラートの可視性を切り替えられるようになりました。	
Manage Profiles から Profile Assignments へのナビゲーションリンクを追加しました。プロファイルの管理からプロファイルの割り当てへのナビゲーションリンクを追加し、管理者がプロファイル構成と割り当てタスクを切り替える必要がある場合のワークフローの効率を向上させます。	
OAuth2 認証を使用した Webhook テスト接続のサポートを追加しました。 OAuth2 認証を使用した Webhook テスト接続のサポートを追加し、ユーザーは OAuth2 資格情報が構成されているときに Webhook エンドポイントの接続を検証できます。	

次のページに続く

表 2 – 前のページからの続き

Description	Addresses
エージェント WebClient 操作の TLS バージョンを指定する構成オプションを追加しました。エージェントでのさまざまな Web クライアント操作の TLS バージョンを指定する構成オプションを追加し、管理者は特定の TLS バージョンを適用してセキュリティコンプライアンスを強化できます。	
PEM 10.0 でドロップされた `ENABLE_DATA_ACCESS_TOOLS`および `ENABLE_DEBUGGER`設定を再導入しました。 PEM 10.0 リリースで削除された `ENABLE_DATA_ACCESS_TOOLS`および `ENABLE_DEBUGGER`構成設定デフォルト `True`を再導入しました。	
バックエンド全体で `session_info` CPU サンプリングをバッチ化して、セッションごとの収集オーバーヘッドを削減します。すべてのバックエンドでバッチ化された `session_info` CPU サンプリングにより、多くのアクティブな接続があるシステムでのセッションごとの CPU 使用量収集のオーバーヘッドが削減されます。	59499
CTE を使用して `probe_target_view` SQL を最適化し、不要なジョインを削除して、パフォーマンスを向上させます。 CTE を使用するように `probe_target_view`の SQL を最適化し、データベース以下のプローブターゲットレベルの不要な結合を削除し、クエリーのパフォーマンスを向上させました。	
年齢ベースと古いオブジェクトのパージを 1 つの機能に組み合わせることにより、パージジョブを最適化しました。経過期間ベースと古いオブジェクトのパージを単一の機能に組み合わせることにより、パージジョブを最適化し、データのクリーンアップに必要なデータベーススキャンの数を削減します。	56990

2.1.3 セキュリティ修正

Description	Addresses
メンテナンスツールオプションの SQL インジェクションの脆弱性 CVE-2026-7815 を修正しました。メンテナンスツールオプションの SQL インジェクションの脆弱性 CVE-2026-7815 を修正しました。*vacuum_index_cleanup*、*vacuum_parallel*、および `buffer_usage_limit`の安全でない値が SQL テンプレートに直接レンダリングされます。サーバーサイドの許可リスト検証とテーブルスペース識別子の `qtIdent`エスケープが追加されました。	
Import/Export クエリエクスポートの OS コマンドインジェクションの脆弱性 CVE-2026-7816 を修正しました。 `psql`テンプレートに渡されるサニタイズされていないクエリコンテンツとフォーマットオプションを介して、インポート/エクスポートクエリのエクスポートにおける OS コマンドインジェクションの脆弱性 CVE-2026-7816 を修正しました。ホワイトリスト検証と平衡括弧クエリパーサーが追加されました。	

次のページに続く

表 3 – 前のページからの続き

Description	Addresses
セッションマネージャーのピクルスデシリアライゼーション RCE 脆弱性 CVE-2026-7818 を修正しました。セッションマネージャーのピクルス逆シリアル化リモートコード実行の脆弱性 CVE-2026-7818 を修正し、細工されたセッションファイルが PEM プロセスで任意のコードを実行できる可能性があります。セッションファイルは、逆シリアル化の前に `hmac.compare_digest` で HMAC 検証されるようになりました。	
ファイルマネージャーのシンボリックリンクパストラバーサルの脆弱性 CVE-2026-7819 を修正しました。ファイルマネージャーのシンボリックリンクパストラバーサルの脆弱性 CVE-2026-7819 を修正し、認証されたユーザーが自分のストレージディレクトリの外部にファイルを書き込むことができました。 `check_access_permission` は `os.path.realpath` を使用し、ファイルのアップロードは `O_NOFOLLOW` を使用するようになりました。	
ダイアログ COMMENT ステートメントと pgstattuple の SQL インジェクションの脆弱性 CVE-2026-12044 を修正しました。 `qtLiteral` エスケープフィルタに切り替え、`conn` が falsy の場合にサイレントに生の値を返すように `qtLiteral` を強化することにより、`COMMENT ON ... IS '<description>'` ステートメントをレンダリングするダイアログテンプレートにわたる SQL インジェクションの脆弱性 CVE-2026-12044 を修正しました。また、 pgstattuple / pgstatindex シンクもクローズしました。これにより、低特権ユーザーがアポストロフィを含む識別子でテーブルまたはインデックスを命名すると、スーパーユーザービューアのロールで SQL インジェクションが発生しました。 Jasser Chebbi がレポートします。	
任意の SQL の実行を可能にする AI Assistant 読み取り専用トランザクションバイパスの脆弱性 CVE-2026-12045 を修正しました。 AI アシスタントの読み取り専用トランザクションバイパス脆弱性 CVE-2026-12045 を修正しました。プロンプト挿入されたマルチステートメントペイロードが `READ ONLY` ラッパーからコミットし、任意の SQL を実行し、スーパーユーザー接続で `COPY ... TO PROGRAM` を介してリモートコード実行にチェーンすることができます。アイザック・チェンによるレポート。	
逆シリアル化シンクを公開した 2 つの SQL Editor エンドポイント認証の欠落 CVE-2026-12046 を修正しました。 `@pga_login_required` デコレータ CVE-2026-12046 がない 2 つの SQL Editor エンドポイント `close` および `update_connection` を修正し、サーバーモードで認証なしで到達可能にし、ピクルス逆シリアル化シンクを公開します。フェルナンド・ボルトッティ記者。	
サーバーエラーテキストと Explain コンテンツを介して、保存された XSS 脆弱性 CVE-2026-12048 を修正しました。 PostgreSQL サーバーエラーテキストと Explain プランノードコンテンツを介して、保存されたクロスサイトスクリプティングの脆弱性 CVE-2026-12048 を修正した。 JavaScript は、保存されたサーバー接続資格情報を読み取り、接続されているすべてのサーバーに対して任意の SQL を発行できます。フェルナンド・ボルトッティ記者。	

次のページに続く

表 3 – 前のページからの続き

Description	Addresses
名前付け復元ポイントエンドポイントの SQL インジェクションの脆弱性 CVE-2026-12050 を修正しました。名前付けリストアポイントエンドポイントの SQL インジェクションの脆弱性 CVE-2026-12050 を修正しました。ユーザー指定のリストアポイント名は、バインドされたパラメーターとして渡される代わりに `str.format()`を介して SQL に補間されました。修正は pgAdmin `ServerNode`と PEM `PemServerNode`実装の両方に適用されました。ゲオが報じた。	
ツリーラベルと Explain ビジュアライザーの `innerHTML`を介して、保存型 XSS 脆弱性 CVE-2026-7814 を修正しました。ブラウザツリーノードラベルと Explain ビジュアライザーの `innerHTML`を介して、保存されたクロスサイトスクリプティングの脆弱性 CVE-2026-7814 を修正しました。すべての `innerHTML`割り当ては `textContent`に置き換えられました。	
Flask-Security */login* `**エンドポイントを介してアカウントロックアウトバイパス脆弱性** *CVE-2026-7820* **を修正しました。アカウントロックアウトバイパスの脆弱性** *CVE-2026-7820* **を修正し、**/authenticate/login`でロックアウトしきい値に達した攻撃者は、Flask-Security `/login`エンドポイントを介してブルートフォースを継続する可能性があります。 `User.is_active`および`User.is_locked`は、両方の認証パスをゲートするようになりました。	
クラウド展開モジュールの HTML インジェクションの脆弱性 CVE-2026-12047 を修正しました。クラウド展開モジュール RDS、Azure、Google の HTML インジェクションの脆弱性 CVE-2026-12047 を修正しました。ここで、SDK 例外テキストはサニタイズされずにブラウザーに転送され、クラウドウィザードの `html-react-parser`を介してレンダリングされます。フェルナンド・ボルトッティ記者。	
多要素認証フローのオープンリダイレクトの脆弱性 CVE-2026-12049 を修正しました。未検証の'next' パラメーターを介して、多要素認証フローのオープンリダイレクトの脆弱性 CVE-2026-12049 を修正しました。細工されたリンクが認証された被害者を攻撃者制御ホストにリダイレクトできます。 Mai Pham Hien 氏によるレポート。	
dompurify を 3.4.8 に更新して、BDSA-2026-13267 を修正しました。 dompurify を 3.4.8 に更新して、 BDSA-2026-13267 を修正しました。	
PEM サーバーインストーラーの HTTPD をバージョン 2.4.67 にアップグレードしました。 PEM サーバーインストーラーの HTTPD をバージョン 2.4.67 にアップグレードし、最新のセキュリティ修正と安定性の向上を組み込みました。	
Windows の PEM サーバーインストーラー用に、バンドルされた PostgreSQL をバージョン 17.10-1 にアップグレードしました。 Windows の PEM サーバーインストーラー用にバンドルの PostgreSQL をバージョン 17.10-1 にアップグレードし、最新のセキュリティ修正と安定性の向上を組み込みました。	
最小限の pgAdmin オリジンコードチャーンで 18 の Dependabot セキュリティアラートをクローズしました。最小限の pgAdmin オリジンコードチャーンで 18 の Dependabot セキュリティアラートをクローズしました。	

2.1.4 の変更

Description	Addresses
REST API v17 とサーバーサイド処理を使用して、ファーストクラスの PEM オブジェクトにアラートのブラックアウトがリワークされました。ジョブベースのアーティファクトから、正規化されたスキーマ、REST API v17 エンドポイント、監査証跡、手動有効/無効トグル、および従来のエージェントサイドを置き換えるサーバーサイドの `process_blackouts()`システムジョブを備えたファーストクラスの PEM オブジェクトへのアラートブラックアウトブラックアウト管理。	
コア使用状況レポートをカスタムレポートテンプレートフレームワークに移行しました。コア使用状況レポートをカスタムレポートテンプレートフレームワークに移行し、スケジュールされた実行、API アクセス、および他のカスタムレポートと合わせた一貫した管理が有効になります。	
システム構成レポートをカスタムレポートテンプレートフレームワークに移行しました。システム構成レポートをカスタムレポートテンプレートフレームワークに移行し、統一されたテンプレートベースのアーキテクチャの下ですべてのシステムレポートを統合しました。	
アラートディスパッチをキュー駆動モデルにリファクタリングし、競合なしのコンカレントワーカーを有効にします。 `FOR UPDATE SKIP LOCKED`コンカレンシーで `pem.alert_queue`を使用して、サイクルごとのテンプレートスキャンからキュー駆動モデルにアラートディスパッチをリファクタリングし、競合なしでコンカレントアラートワーカーを有効にし、ブラックアウト中にドロップするのではなく `due_at`を介してキューに入れられたアラートを遅延します。	
`probe_target_view`マテリアライズドビューを同時に更新して、リフレッシュ中のテーブルロックを排除しました。 `probe_target_view`マテリアライズドビューを同時にリフレッシュするように `pem.refresh_stale_probe_view`ファンクションを変更し、リフレッシュジョブ中のテーブルロックを排除し、負荷時のシステムの応答性を向上させました。	
アップストリームのバグ修正と改善を加えて、SNMP ++をバージョン 3.6.6 にアップグレードしました。 SNMP++をバージョン 3.6.6 にアップグレードし、アップストリームのバグ修正と SNMP 通知サブシステムの改善を組み込みました。	
エージェント接続、スプーラー、および非同期クエリパスを強化して、ハングやサイレント障害を排除します。エージェント接続、スプーラー、および非同期クエリパスを強化して、ハング、ポイズニング行ループ、およびサイレント障害モードを排除します。タイムアウトとキャンセルをサポートする `poll()`駆動の非同期クエリ実行、共有 `SpoolerLoop`ドライバー、制限付きスプール再試行、および型指定をサポート `ErrorKind`カテゴリ。	
プールの枯渇を削減するために、デフォルトの `PEM_DB_POOL_MAX_WAITING`を 10 から 30 に増加しました。デフォルトの `PEM_DB_POOL_MAX_WAITING`を 10 から 30 に増加し、拒否される前にキューに入れる要求を許可することにより、同時負荷の下での接続プールの消耗を削減します。	

次のページに続く

表 4 – 前のページからの続き

Description	Addresses
`pemWorker`のコマンドライン引数解析フレームワークを最新化しました。`pemWorker`のコマンドライン引数解析フレームワークを最新化し、保守性を向上させ、新しい CLI オプションのよりクリーンな追加を可能にします。	
PEM ソースファイル全体で著作権表示を 2026 に更新しました。 PEM ソースファイル全体で著作権表示を 2026 に更新しました。	
pgAdmin 9.13 コードベースを PEM にマージし、最新の機能と修正を提供します。 pgAdmin 9.13 コードベースを PEM にマージし、最新の pgAdmin 機能、バグ修正、および改善を PEM ユーザーインターフェイスとコア機能にもたらします。	
Django webhook_mtls_app を 4.2.26 LTS に更新しました。 Django webhook_mtls_app を 4.2.26 LTS に更新しました。	
urllib3 を 2.7.0 に更新しました。 urllib3 を 2.7.0 に更新しました。	
axios を 1.18.0 に更新しました。 axios を 1.18.0 に更新しました。	
React を 19.2.6 に更新しました。 React を 19.2.6 に更新しました。	
qs を 6.15.2 に更新しました。 qs を 6.15.2 に更新しました。	
lodash を 4.18.1 に更新しました。 lodash を 4.18.1 に更新しました。	
postcss を 8.5.15 に更新しました。 postcss を 8.5.15 に更新しました。	
Authlib を 1.6.12 に更新しました。 Authlib を 1.6.12 に更新しました。	
フォローリダイレクトを 1.16.0 に更新しました。フォローリダイレクトを 1.16.0 に更新しました。	
form-data を 4.0.6 に更新しました。 form-data を 4.0.6 に更新しました。	

2.1.5 バグ修正

Description	Addresses
ASB キャッシュを無効にすることによるローテーション後に、古いキャッシュされた資格情報を使用するエージェントを修正しました。無効な資格情報が原因で監視対象サーバー接続が失敗したときに、エージェントのエージェントサーバーバインディング ASB キャッシュが無効にされず、資格情報のローテーション後にエージェントが古いキャッシュされたパスワードを使用し続けてしまう問題を修正しました。エージェントは、接続障害が発生すると ASB キャッシュを削除し、プーラー資格情報をクリアし、`EDBPEMDecrypt`が失敗した場合にキャッシュされた平文へのフォールバックを拒否するようになりました。	
重複したタイトルでサイレントに失敗する`create_system_report`を修正し、冪等のアップグレードの再実行が可能になりました。 `create_system_report`は、例外を発生する代わりに、重複したタイトルでサイレントに失敗し、既にアップグレードされたインストールでアップグレードスクリプトの冪等の再実行を有効にする問題を修正しました。	

次のページに続く

表 5 – 前のページからの続き

Description	Addresses
[アラートの管理] パネルでアラートパラメーター値を編集する際の構文エラーを修正しました。 [Manage Alerts] パネルからアラートのパラメーター値を編集するときに発生した構文エラーを修正しました。	59918
`snmp_spool`テーブルがパージされなかった `purge_snmp_spool`のタイプミスを修正しました。 `snmp_spool`テーブルが適切にパージされず、未送信の SNMP 通知の蓄積を発生させた `purge_snmp_spool`ファンクションのタイプミスを修正しました。	
アラートキューからサイレントにフィルタリングされるグローバルエージェントサーバーレベルのアラートを修正しました。適格性述語が `agent_id`での `agent_server_binding`一致を誤って必要としたため、グローバルエージェントサーバー固有のアラート *agent_id* = 0 または-1 がアラートキューからサイレントにフィルタリングされ、これにより、自動作成サーバーの約 66%が発生した問題を修正しました。レベルアラートは決して起動されません。	
コピーアラートのクラッシュと、ダブルエンコードされた `connection_params`からの 10.3.1-to-10.4.1 アップグレードの失敗を修正しました。非正規の `package_name`/ *agent_id* **値を持つフィクスチャを除外するスコープフィルターの強化が原因で発生したコピーアラートクラッシュを修正し、ダブルエンコードされた** *`connection_params* JSONB 文字列が `jsonb_each`に移行を中止させる 10.3.1 から 10.4.1 へのアップグレードの失敗を修正しました。	61992
Barman ダッシュボードのローソク足チャートのサーバーフィルタリングを修正しました。 Barman ダッシュボードのローソクチャートでサーバーでフィルタリングできない問題を修正し、サーバーレベルのフィルタリングを復元します。	60789
`pemserver.sql`の重複したファンクションエントリーを修正し、EFM アラートテンプレート名を修正しました。 `pemserver.sql`の `pem.create_delete_probe_purge_jobstep()`ファンクションの重複エントリを修正し、10.3.0 アップグレードスクリプトのアラートテンプレート名「EFM Fewer Than N Nodes Active」を修正しました。	
`pem_assign_profile`の動作を文書化されたプロファイル割り当てコントラクトと調整しました。 `pem_assign_profile`を文書化されたコントラクトと調整し、実際の動作と予想されるプロファイル割り当てセマンティクス間の不一致を修正しました。	
切断されたサーバー状態とヘッダーレイアウトからのパフォーマンス診断ツールの回復を修正しました。パフォーマンス診断ツールが切断されたサーバー状態から正常に回復しない問題を修正し、レスポンシブヘッダーレイアウトを修正しました。	60789
パンくずリストを介してコンテキストをナビゲートするときに、ダッシュボードチャートが更新されない問題を修正しました。パンくずリストナビゲーションを介してコンテキスト間を移動するときに、ダッシュボードチャートが更新されない問題を修正しました。	60091, 61637

次のページに続く

表 5 – 前のページからの続き

Description	Addresses
モデルの誤使用が原因でクエリツールから失敗するワークスペースのアドホックデータベース接続を修正しました。ワークスペースのアドホックデータベース接続が PEM サーバーモデルを正しく使用せず、クエリツールからの接続障害を発生させた問題を修正しました。	
間違ったソースからトランザクション ID を読み取るチャートの色設定 API を修正しました。トランザクション ID がダッシュボードの状態からではなくプロップから読み取られたため、チャートの色設定 API が失敗した問題を修正しました。	
ファイルアクセスエラーを解決するために、 `pg_read_server_files`および `pg_write_server_files`ロールを `pem_agent`に付与しました。`pg_read_server_files`および`pg_write_server_files`ロールを`pem_agent`に付与し、エージェントがモニタリング操作のためにサーバーサイドのファイルにアクセスする必要がある場合の権限エラーを解決します。	
一部のコンテキストでジョブロギング障害を発生させた非修飾の `insert_jobinfolog`ファンクションを修正しました。 `insert_jobinfolog`ファンクションがスキーマ修飾されていないため、特定の実行コンテキストでジョブログの障害が発生する問題を修正しました。	
ログエントリをまとめて実行した改行文字の欠落している PEM agent.log 出力を修正しました。 PEM agent.log 出力に改行文字が含まれていないため、ログエントリが一緒に実行され、ログ分析が困難になる問題を修正しました。	60923
新しいエージェントを登録する際の偽の警告メッセージを削除しました。新しいエージェントを登録するときに表示される偽の警告メッセージを排除し、登録ワークフローのノイズを削減します。	
一時的な接続損失後のプローブとジョブのロック解除操作のインプロセス再試行キューを追加しました。一時的な PEM 接続損失のために失敗したプローブとジョブのロック解除操作用のインプロセス再試行キューを追加し、次のワーカーが再起動するまでプローブが飛行中にスタックしたままになるのを防ぎます。	
UI で削除されたオブジェクトに対して作成されない `purge_deleted_objects_data`ジョブを修正しました。 UI から削除されたオブジェクトに対して `purge_deleted_objects_data`ジョブが作成されない問題を修正し、`pemdata`スキーマからデータを削除する前にトリガーを無効にするようにパージファンクションを変更して、パージ中に `pemhistory`へのコピーを防止しました。	
`/agent/job`エンドポイントの API ドキュメントと検証を修正しました。`/agent/job`エンドポイントの API ドキュメントと検証を修正し、正しい要求と応答スキーマを保証します。	
サービスの再起動中にコアダンプを生成する `edb-uwsgi`を修正しました。プロセスのライフサイクル処理を修正することにより、サービスの再起動中に `edb-uwsgi`がコアダンプを生成した問題を修正しました。	

次のページに続く

表 5 – 前のページからの続き

Description	Addresses
アップグレードスクリプトが失敗したときにエラーを出力するように PEM アップグレードを修正し、部分的なアップグレードを防止します。 PEM のアップグレードを修正して、アップグレードスクリプトが失敗したときにエラーを出力するようにし、システムを一貫性のない状態のままにするサイレント部分アップグレードを防ぎます。	
実際のホスト名と HTTPS スキームをアプリケーションに渡すように Nginx を修正しました。実際のホスト名と HTTPS スキームが uWSGI /Flask アプリケーションにパススルーされるように Nginx 構成を修正し、リバースプロキシの背後での URL 生成とリダイレクト処理に関する問題を解決しました。	
バックエンドの再起動後に修復するように PEM データベース接続プールを修正しました。バックエンドの再起動後に修復するように PEM データベース接続プールを修正し、プールレベルのヘルスチェックと古い接続の適切な処理を追加することにより、Web UI での `NoneType`のクラッシュを防止します。	
継続的なレンダリングのために空のタイムラインバケットをバックフィルするようにパフォーマンス診断チャートを修正しました。空のタイムラインバケットがバックフィルされるようにパフォーマンス診断チャートを修正し、一部の期間にデータがない場合でも継続的なチャートレンダリングを保証します。	
プローブ REST API を強化し、Create Profile ダイアログの重複名前検証エラーを修正しました。プローブ REST API システムプローブでの SQL 編集のブロック、POST での重複の `probe_name`の拒否、およびエクスポート時に不明な ID の表面化を強化し、Create Profile ダイアログを修正して、生の PostgreSQL 一意コンストレインの代わりに重複したプロファイル名のインライン検証エラーを表示しますエラー。	
アラートの詳細チャートでアラートを復元しない「Showacknowledgmentalerts」を修正しました。データが正常に返されたときにエラー状態がクリアされなかったため、すべてのアラートが確認された後、[Showacknowledalerts] を有効にしてもアラートの詳細チャートにアラートが復元されない問題を修正しました。	
ダーク、ライト、ハイコントラストのテーマ全体で WCAG 2.0 AA コントラストの問題を修正しました。ダーク、ライト、ハイコントラストのテーマ全体で WCAG 2.0 AA コントラストの問題を修正し、視覚障害を持つユーザーのアクセシビリティを向上させます。	
カスタムバッチプローブのプラットフォームを編集しても、データベースプラットフォーム列が更新されない問題を修正しました。カスタムバッチプローブのプラットフォームの編集たとえば、*nix から Windows に変更すると、データベースのプラットフォーム列が更新されず、プローブが目的のエージェントの `probe_target_view`に表示されない問題を修正しました。	

次のページに続く

表 5 – 前のページからの続き

Description	Addresses
pem_manage_alert `ユーザーが *REST API* を介してアラートを作成できないを修正した。 `*pem.webhook_alert_config* `テーブルに `*SELECT*、*INSERT*、*UPDATE*、および `DELETE`権限がないため、`pem_manage_alert`ロール `pem_config_alert`を介してユーザーが REST API を介してアラートを作成できない問題を修正しました。	
データベースレベルのプローブで失敗する `--disable-probe`および `--enable-probe` CLI コマンドを修正しました。 *--disable-probe* `および`--enable-probe` CLI コマンドがデータベースレベルのプローブで失敗し、「permission denied for table *probe_config_database*」エラーを修正しました。	

2.2 Postgres Enterprise Manager 10.4.2 release notes

リリース日 2026 年 5 月 14 日

Postgres Enterprise Manager 10.4.2 のバグ修正とセキュリティの向上には次のものが含まれます。

2.2.1 セキュリティ修正

Description	Addresses
修正した CVE-2026-7814 — `innerHTML`ではなく `textContent`を介して細工された PostgreSQL オブジェクト名をレンダリングすることにより、ブラウザツリーに保存される XSS を防止しました pgAdmin commit 3294e74 から移植。	
修正 CVE-2026-7815 — *INDEX_CLEANUP*、*PARALLEL*、*BUFFER_USAGE_LIMIT*、および `TABLESPACE`オプションフィールドを許可リスト検証し、*reindex_tablespace*～*qtIdent* pgAdmin commit cf53953d9 から移植することをルーティングすることにより、メンテナンスツールでの SQL インジェクションを修正しました。	
修正 CVE-2026-7816 — psql-strtokx モデルのかっこバランスパーサーの導入により、インポート/エクスポートクエリエクスポート機能での OS コマンドインジェクションを修正し、null バイトの拒否、改行の正規化、および`format`、`on_error`のホワイトリスト検証、および`log_verbosity`パラメーター pgAdmin commit 13badc62c から移植。	
修正 CVE-2026-7818 — シリアル化された本文に SHA-256 HMAC を先頭に追加し、逆シリアル化の前に `hmac.compare_digest`で検証することにより、ファイルバッキングセッションマネージャーでの安全でないピクルスデシリアライズを修正しました。以前のバージョンで書き込まれた既存のセッションファイルは、読み取り時にサイレントに無効化されます。ユーザーはアップグレード後に 1 回再認証が必要です pgAdmin commit 30a890337 から移植。	

次のページに続く

表 6 – 前のページからの続き

Description	Addresses
修正 CVE-2026-7819 — `check_access_permission`の `os.path.realpath`を介したパスを解決し、`O_NOFOLLOW`および `0o600`ファイルモードでアップロードを開くことにより、ファイルマネージャーのシンボリックリンクパストラバーサルを修正しました pgAdmin commit 435752b83 から移植。	
修正 CVE-2026-7820 — すべての認証パスの `locked`列を参照するように `User.is_active`および `User.is_locked()`をオーバーライドすることにより、Flask-Security のデフォルトの`/login`エンドポイントを介したアカウントロックアウトバイパスを修正しました。 SQLite 専用のデータ正規化移行が含まれます。 PostgreSQL 構成データベースの no-op pgAdmin commit d336c1e78 から移植。	
暗号化を 46.0.7 に更新して CVE-2026-39892 を修正し、暗号化操作の潜在的なセキュリティ脆弱性からユーザーを保護します。	
urllib3 を 2.7.0 に更新して BDSA-2026-9851 を修正し、HTTP クライアント操作の潜在的なセキュリティ脆弱性からユーザーを保護します。	
Authlib を 1.6.12 に更新して CVE-2026-41425 を修正し、OAuth および JWT 処理の潜在的なセキュリティ脆弱性からユーザーを保護します。	
Mako を 1.3.12 に更新して CVE-2026-41205 を修正し、テンプレートレンダリングの潜在的なセキュリティ脆弱性からユーザーを保護します。	
axios/follow-redirects を 1.16.0 に更新して、CVE-2026-40175、CVE-2026-42033、および CVE-2026-42035 を修正し、HTTP リダイレクト処理の潜在的なセキュリティ脆弱性からユーザーを保護します。	
PostCSS を 8.5.14 に更新して CVE-2026-41305 を修正し、CSS 処理の潜在的なセキュリティ脆弱性からユーザーを保護します。	

2.2.2 バグ修正

Description	Addresses
アップグレードスクリプトが失敗すると PEM アップグレードがエラーを発生し、サイレント部分アップグレードを防止します。	
React を 19.2.6 に更新して CVE-2026-23869 を修正し、UI レンダリングレイヤーの潜在的なセキュリティ脆弱性からユーザーを保護します。	
log_connections 列の `*pemhistory* `log_configuration`テーブルを処理するようにアップグレードファイルを修正しました。	
Barman ダッシュボードのローソク足チャートでサーバーでフィルタリングできない問題を修正しました。	

2.3 Postgres Enterprise Manager 10.4.1 release notes

リリース日 2026 年 4 月 9 日

Postgres Enterprise Manager 10.4.1 の新機能、機能強化、バグ修正、およびその他の変更には次のものが含まれます。

2.3.1 セキュリティ修正

Description	Addresses
暗号化を 46.0.6 に更新して、CVE-2026-34073 を修正しました。	
要求を 2.33.1 に更新して、CVE-2026-25645 を修正しました。	
Pygments を 2.20.0 に更新して、CVE-2026-4539 を修正しました。	
ajv を 8.18.0 に更新して、CVE-2025-69873 を修正しました。	
Lodash を 4.18.1 に更新して、CVE-2026-4800 および CVE-2026-2950 を修正しました。	

2.3.2 バグ修正

Description	Addresses
親ノードを選択すると、そのすべての子ノードが選択される問題が修正されました。	
Audit / Probe / Server ログダッシュボードのスクロールの問題を修正しました。	
pem_admin ロールを持つユーザーがチームアクセスに関係なくすべてのサーバーを表示できる問題を修正しました。	

2.4 Postgres Enterprise Manager 10.4.0 release notes

リリース日 2026 年 3 月 20 日

Postgres Enterprise Manager 10.4 の新機能、機能強化、バグ修正、およびその他の変更には次のものが含まれます。

2.4.1 ハイライト

- 監視対象サーバーまたは PEM サーバーで実行できる新しいユーザー定義レポート。
- より柔軟なインストールを可能にする新しい `configure-pem-server.sh` オプション。
- 新しい構成オプションを使用した HA PEM 展開の完全に改訂されたドキュメント。
- SLES 15 SP7 のサポート。

2.4.2 機能強化

Description	Addresses
包括的なカスタムレポート機能を導入し、ユーザーが PEM サーバーまたは監視対象サーバーでユーザー定義のレポートテンプレートを作成、実行、および管理できます。スケジュールされたジョブとしてのレポート実行と PEM API との統合がサポートされています。	
`configure-pem-server.sh`の新しいオプションを導入して、より柔軟な PEM 展開を有効にします。これには、ホストエージェントのパススルーオプションを提供し、実行中の HA Postgres クラスターのプライマリノードとスタンバイノードでスクリプトを実行する機能が含まれます。	
削除プロセス中にアラートプロファイルをあるターゲットから別のターゲットに移動できるプロファイルの再割り当て機能を実装しました。これは、プロファイルを単に削除するというデフォルトの動作をオーバーライドし、特定のプロファイルが削除されても監視構成がそのまま残るようにします。	
pemWorker エージェント接続の接続文字列サポートを実装し、ユーザーが単一の文字列ですべての接続パラメーターを指定できることにより展開を簡素化し、個別の SSL 証明書構成の必要性を削減します。	
<details><summary>Webhook エンドポイントの OAuth 2.0 クライアント資格情報 M2M 認証の暫定的なサポートを追加しました。</summary> OAuth を使用するように Webhook を構成できるようになりました。最初にトークンを取得し、次に使用しますそのトークンを使用して後続の HTTP 要求を認証します。サポートは暫定的なものとしてマークされています。つまり、実稼働使用に適していますが、将来のマイナーリリースで重大な変更が加えられる可能性があります。 </details>	
pemworker の新しい `verify-registration`コマンドを追加して、ユーザーがエージェントが PEM サーバーに適切に登録されているかどうかを確認できるため、ログファイルを検査せずにエージェント登録ステータスの独立した検証を有効にします。	
古いイベント履歴レコードを自動的にクリーンアップするスケジュールされたシステムジョブを追加し、ユーザーがデータベースの成長を管理し、時間の経過とともに最適なパフォーマンスを維持できるようにします。	
プロファイル管理用の包括的な REST API v16 を導入し、開発者が標準の API エンドポイントを介してアラートプロファイルをプログラム的に作成、取得、更新、および管理できるようにします。	
新しい特権ロールを導入しました。BARMAN サーバーをリストする `pem_list_barman`、 *team* **`フィールドに関係なくすべての監視対象リソース/アセットを表示できる** *`pem_list_all_assets*、データベースサーバー、エージェントまたはツール BARMAN を追加/削除する `pem_manage_assets`、および割り当てるための `pem_assign_profile`プロファイルをエージェント/サーバーに送信します。	
アラートリマインダーメールを無効にするサポートを追加し、ユーザーが通知設定をきめ細かく制御できるようにして、繰り返しのアラート通知によるメールの疲労を軽減します。	
「+」記号で始まるアドレスと長いドメイン名をサポートするようにメールアドレス検証を拡張し、Gmail のプラスアドレス機能のような最新のメールアドレス指定スキームを使用するユーザーに対応します。	

次のページに続く

表 10 – 前のページからの続き

Description	Addresses
SUSE Linux Enterprise Server 15 Service Pack 7 SLES/15/SP7 のサポートを追加しました。	

2.4.3 の変更

Description	Addresses
PEM Web アプリケーションは、Ubuntu 24.04 を除くすべての Linux プラットフォームで edb-python 3.12 インタープリターを使用するようになりました。	
RHEL 9 の PEM パッケージは RHEL 9.6 になったので、パッケージは RHEL 9.6 と 9.7 の両方で動作できます。その結果、PEM 10.3ARM および PEM 10.3.1Intel/ARM を使用しているユーザーは、完全なサポートを維持するには PEM 10.4 にアップグレードする必要があります。	
pgAdmin 9.10 コードベースを PEM にマージし、最新の pgAdmin 機能、バグ修正、および改善を PEM ユーザーインターフェイスとコア機能にもたらします。	

2.4.4 バグ修正

Description	Addresses
Cleared Alerts フラグがコピーされないアラートのコピー機能の問題を修正し、アラートをコピーするときにすべてのアラートプロパティが適切に複製されるようにします。	41904
qtliteral ファンクションを修正して、エスケープ処理を psycopg ライブラリに委任することにより SQL フォーマットの問題を適切に処理し、手動引用符の二重化バグを排除します。	55011, 55481
`log_configuration`テーブルで `log_connections`値を'true' / 'false' から標準の PostgreSQL 形式の'on'/'off' に移行し、PostgreSQL 構成標準との一貫性を確保しました。	55651
テーブルスペースディレクトリが欠落しているためにシステム構成レポートがエラーをスローしていた問題を修正しました。また、このようなゴーストテーブルスペースについてユーザーに積極的に通知するために、アラートテンプレート「Tablespace Directory Missing」も作成しました。	53777
エージェント SSL 証明書とキーが `--use-agent-sslcrt`および `--use-agent-sslkey`を介して指定される場合、不必要な証明書パスのチェックと作成を修正し、既存の証明書を使用したエージェント登録を合理化します。	56476
データベースのタイムゾーンが非 UTC タイムゾーンである場合、 'alerting resume' が誤った将来の時刻を表示する問題を修正しました。	57785
`CONFIG_DATABASE_URI`を介してユーザー設定のデータベースとして PG / EPAS を使用する場合のログインエラーを修正しました。	57193

次のページに続く

表 12 – 前のページからの続き

Description	Addresses
urllib3 をバージョン 2.6.3 に更新して CVE-2026-21441 を修正し、HTTP クライアント操作の潜在的なセキュリティ脆弱性からユーザーを保護します。	
require.js バージョンを 2.3.6 から 2.3.8 にアップし、上流のバグ修正と JavaScript モジュールローダーのセキュリティの向上を組み込みました。	
同じターゲットレベルの重複したチャート名が誤って許可される問題を修正し、チャートの一意性を保証し、ダッシュボードを管理する際のユーザーの混乱を防ぎます。	
電子メールテンプレートをリセットすると、実際のリセット値を反映する代わりに古いデータが返される問題を修正し、リセットを実行した後にユーザーに正確なテンプレートコンテンツが表示されるようにします。	
Transaction ID Exhaustion Wraparound プローブを修正して、値を整数ではなく浮動小数点にキャストし、データのオーバーフローを防止し、トランザクション ID 使用状況モニタリングの正確なパーセンテージ計算を提供します。	
イベント履歴ロギングと UI 通知に影響を与えるプロファイル草案ロジックの問題を解決し、プロファイルの変更が正確に追跡され、ユーザーがプロファイル管理中に適切なフィードバックを受け取ることを保証します。	
auto_created 列を profile_alert_configs テーブルに追加し、削除ロジックを更新して、プロファイルからシステム作成のアラートが誤って削除されるのを防ぎ、重要な監視構成を保護します。	
boost::condition_variable を使用して最新の C++コンストラクトを使用して、pemworker での監視対象サーバーの接続の割り当てを改善し、スレッドの安全性を強化し、接続割り当て中の CPU 使用率を削減します。	
登録中にエージェント登録オプション `-o log_level=xxx`および `-o log_location=yyy`が無視される問題を修正し、ユーザーがエージェントのセットアップ中にロギング構成を適切にカスタマイズできるようにします。	
適切なロールベースの制限を適用するようにサーバー PUT API を変更し、databaseServerRegistration ロールを持たないユーザーがサーバープロパティを編集できないようにし、機密接続パラメーターを暗号化することによりセキュリティを強化しました。	
サーバー登録の SSL エラーを修正し、サーバー挿入ステートメントのスキーマワイズマッピングを使用するようにサーバー登録ロジックをリファクタリングし、コードの保守性と登録の信頼性を向上させました。	
すべての更新ではなく、`server_version_id`列が更新されたときにのみマテリアライズドビューをリフレッシュするように`trg_server_info_flag_mv`トリガーを最適化し、不必要なデータベースのオーバーヘッドを大幅に削減します。	
pemdata.server_info テーブルに基づいて、pem_host_and_server テーブルの is_leader フラグを更新するトリガーを追加しました。	

2.5 Postgres Enterprise Manager 10.3.1 release notes

リリース日 2026 年 2 月 25 日

!!!重要 PEM エージェント 10.3.1 互換性と登録 PEM エージェント 10.3.1 の `pemworker` ユーティリティは、PEM サーバーバージョンが 10.3 より前の場合、新しいサーバーを登録できません。これは既知の問題で

あり、将来のリリースで解決が予定されています。

推奨される回避策 _ 10.3 より前の PEM サーバーバージョンにサーバーを登録する必要がある場合、次のいずれかの方法を使用できます。

1. **Web UI を介した登録** _ PEM Web アプリケーションインターフェイスを使用してサーバーを登録します。
2. **段階的なエージェントのアップグレード** _ 最初に PEM エージェントバージョン 10.2 を使用してサーバーを登録し、次にエージェントをバージョン 10.3.1 にアップグレードします。 .. :: これはセキュリティパッチであり、すべての PEM 10 ユーザーにお勧めします。

2.5.1 の変更

Description	Addresses
PEM は、RHEL 8 x86 の OS 提供の libcurl パッケージに依存するようになりました。これはすでに他のすべてのプラットフォームに当てはまります。非推奨の<code>libcurl-pem</code>パッケージは、アップグレード後に安全に削除できます。	

2.5.2 バグ修正

Description	Addresses
暗号化が 46.0.5 修正 CVE-2026-26007 に更新されました。	
urllib3 が 2.6.3 修正 CVE-2025-66418、CVE-2025-66471、CVE-2026-21441 に更新されました。	
Authlib は 1.6.7 修正 CVE-2025-68158 に更新されました。	
Pillow は 12.1.1 修正 CVE-2026-25990、Python 3.10 以降を使用するプラットフォームの場合。	
Werkzeug は 3.1.5 修正 CVE-2025-66221、CVE-2026-21860 に更新されました。	
PyNaCl は 1.6.2 修正 CVE-2026-26007 に更新されました。	
pyasn1 は 0.6.2 修正 CVE-2026-23490 に更新されました。	
RequireJS は 2.3.8 修正 CVE-2024-38999 に更新されました。	
Swagger-UI は 5.31.0 修正 CVE-2021-46708、CVE-2018-25031 に更新されました。	
Axios は 1.13.5 修正 CVE-2025-27152、CVE-2026-25639、CVE-2025-58754、CVE-2024-57965 に更新されました。	
プレーン SQL のリストアは、有害な psql メタコマンドを防ぐために'restrict'オプションを使用して実行されるようになりました CVE-2025-13780 を修正します。	
PEM は、プレーン SQL ファイルを復元するときにプロセスウォッチャーの'restrict' オプションの秘密キーをマスクするようになりました CVE-2026-1707 を修正します。	

2.6 Postgres Enterprise Manager 10.3.0 release notes

リリース日 2025 年 11 月 25 日

!!!重要 `libcurl-minimal` パッケージがインストールされている RHEL 9 システムで PEM 10.3 にアップグレードする場合、`dnf --allow-erasing` オプションを渡す必要があります。

Postgres Enterprise Manager 10.3 の新機能、機能強化、バグ修正、およびその他の変更には次のものが含まれます。

2.6.1 ハイライト

- 多くのサーバーとエージェントにわたるプローブとアラート構成の管理を簡単にする新しい**プロファイル** 機能
- PEM を使用した独自の SSL インフラストラクチャの導入を簡単にする新しいオプション
- EFM クラスターの状態に関する新しいビルトインアラート
- FIPS 準拠の暗号化モジュール
- Postgres 18 のサポート

2.6.2 機能強化

Description	Addresses
PostgreSQL、 EDB Postgres Extended Server、および EDB Postgres Advanced Server のサポートを追加しました。これらの Postgres バージョンは、PEM バックエンドサーバーとして監視したり、使用したりできます。	
PEM の包括的なプロファイル管理機能を実装し、管理者は、事前定義されたプローブとアラート構成を含むプロファイルを作成、管理、および複数のエージェントとサーバーに割り当てることができます。新しいプロファイル管理機能を使用すると、ユーザーは事前定義されたプローブとアラート構成を使用してプロファイルを作成、管理、および複数のエージェントとサーバーに割り当て、監視セットアップを合理化できます。	
クエリツールやスキーマ差分ツールなどのさまざまな pgAdmin ベースのツールへのユーザーアクセスを制御するために、新しい詳細な Postgres ロールが導入されました。次の新しいロールが追加されました。 - query_tool - schema_diff_tool - debugger_tool - erd_tool -rant_wizard -backup_tool - restore_tool - import_export_data_tool - search_objects_tool - manage_tool	

次のページに続く

表 15 – 前のページからの続き

Description	Addresses
パスワードストレージ用に FIPS 準拠の SHA256 ベースの PEM 暗号化モジュールを実装しました。保存されたパスワードの暗号化に SHA256 を使用した新しい暗号化モジュールを実装し、古い SHA1 ベースの暗号化を非推奨にします。 PEM 10.3 の新規インストールは、新しいモジュールのみを使用します。10.3 にアップグレードされたシステムは、新しいモジュールを使用してパスワードを暗号化し、古いモジュールを使用して既存のパスワードを復号化します。 PEM の将来のリリースでは、既存のパスワードを新しいモジュールに移行するメカニズムを提供します。	
アラートテンプレート「EFM Fewer Than N Nodes Active」を追加しました。EFM クラスター内のアクティブなノードの数を監視するために、新しいアラートテンプレート「EFM Fewer Than N Nodes Active」が追加されました。新しいアラートテンプレートは自動作成するように設定されていないため、ユーザーは必要に応じてそれらからアラートを作成する必要があります。このアラートを使用するには、サーバーレプリケーションソリューションを EFM に設定する必要があります。	
サーバーおよびエージェントエンドポイントでの`profile_id`割り当てをサポートする新しい API バージョン v15 を導入しました。	
PEM 構成およびエージェント登録中に SSL 証明書の作成を抑制する新しいオプションを追加しました。 SSL 証明書が外部で提供される環境に PEM をインストールする場合、configure_pem_server.sh を呼び出すときにこれらの新しいオプションを使用して、PEM による SSL 証明書の生成を抑制できます。 --skip-db-certificate オプションには次の効果があります。- Postgres のサーバー証明書の作成を防止します。 - sslutilsextension のチェックと CA 証明書の作成を無効にします。および sets-o check_server_certificate_expiry=falseand-o Skip_db_certificate=trueeduring ローカルエージェント登録中。クライアント証明書は生成されず、証明書更新システムジョブは無効になります。 --skip-web-certificate オプションは、自己署名 Web サーバー証明書の作成を防止します。 --use-trusted-ssl-db-certificate オプションは、--skip-db-certificate および-no-hba-change の両方と同等の便利なフラグです。	
PEM 関連パネルのタブの保存を追加しました。 PEM 関連パネルにタブの保持が実装され、アプリケーションをナビゲートまたは再起動しても、開いているタブがアクティブなままになります。	
管理プロファイルのイベント履歴追跡を実装しました。プロファイルの管理機能にイベント履歴追跡が実装され、監査目的ですべてのプロファイル関連の変更を追跡します。	
プロファイルを管理するためのロールを追加しました。プロファイル管理用に、新しいロール pem_manage_profile が追加されました。このロールは、pem_probe_config および pem_alert_config から権限を継承します。	
PEM は、48 時間到達できなかったサーバーまたはエージェントを自動的にブラックアウトします。スケジュールされたジョブは時間ごとに実行され、server_contact_timeout 構成に対してサーバー/エージェントのハートビートを確認します。 48 時間到達できない場合、オブジェクトはブラックアウトされます。再び到達可能な場合、ブラックアウトは元に戻されます。	

次のページに続く

表 15 – 前のページからの続き

Description	Addresses
「EFM Agent Down」および「EFM Missing Primary」のビルトインアラートテンプレートを追加しました。これらの新しいアラートテンプレートは自動作成するように設定されていないため、ユーザーは必要に応じてそれらからアラートを作成する必要があります。これらのアラートを使用するには、サーバーレプリケーションソリューションを EFM に設定する必要があります。	
Windows の PEM インストーラーは、 sslutils 1.4 を統合するようになりました。これにより、パラメーターを介して証明書の有効期限を指定するサポートが追加されます。	
PEM エージェント Windows インストーラーの Boost パッケージを 1.89 にアップグレードしました。	
PEM サーバー Windows インストーラーの Postgresql サーバーパッケージを 17.6-2 にアップグレードしました。	
ジョブのコンテキストログを追加しました。 PEM エージェントジョブにコンテキスト依存ログが追加され、worker.log のトラブルシューティングに必要なより詳細な関連情報を提供します。	

2.6.3 の変更

Description	Addresses
環境変数と競合しないように、 HOST の代わりに変数 PEM_HOST を使用します。 Web モードでの PEM のインストール中に PEM DB サーバーホストアドレスに使用される HOST 変数は、既存の環境変数との競合を回避するために PEM_HOST に変更されました。	
PEM は、--pem-agent-user 指定でエージェントを登録するときに、anagent_<id>user を作成しなくなりました。以前のバージョンでは、既存のユーザーが--pem-agent-user で指定された場合でも、PEM は登録されたすべてのエージェントに対して Postgres ユーザーを作成しました。これは、PEM 10.3 では当てはまりません。 Postgres ユーザーは指定しない場合にのみ作成されます。	43623
パフォーマンス診断選択バーがデフォルトで全幅に設定されるようになりました。パフォーマンス診断では、紫の選択バーがデフォルトで全幅になり、クエリリストがすぐに入力され、使いやすさが向上しました。	
PEM はデフォルトで Barman プローブを有効にするようになりました。以前のバージョンでは、 Barman プローブはデフォルトで無効になっており、インスタンスによって登録されたデータを収集する前に有効にする必要がありました。 Barman プローブは、 Barman インスタンスがない場合は何もせず、すべての Barman 機能に不可欠であるため、PEM の構成時にデフォルトで有効にします。	
SNMP++ライブラリは、以前の 3.6.0 からバージョン 3.6.3 にアップグレードされました。	

次のページに続く

表 16 – 前のページからの続き

Description	Addresses
'System Wait' および'Session Wait' ダッシュボードオプションは、PostgreSQL および EDB Postgres Advanced Server バージョン 18 以降を監視する場合、非表示になります。	
MUI v5 から v7 にアップグレードしました。	

2.6.4 バグ修正

Description	Addresses
--no-hba-change が指定された場合でも、configure-pem-server.sh スクリプトが pg_hba.conf ファイルを上書きする問題を修正しました。既存の HBA エントリをチェックするための厳しすぎる正規表現が原因で、--no-hba-change が指定された場合でも、configure-pem-server.sh スクリプトが pg_hba.conf ファイルを上書きする問題を解決しました。	54064
RHEL 9 に libcurl-minimalis がインストールされている場合、PEM エージェントが電子メールを送信できなかった問題を修正しました。 PEM エージェントは、SMTP をメールサーバーに送信するために libcurl パッケージが必要です。 libcurl のパッケージレベルの要件は、RHEL 9 ビルドで頻繁に見られる packagelibcurl-minimal によって満たされることがわかりました。ただし、このパッケージは SMTP に必要なライブラリを提供していないため、PEM 10.3 は libcurl-full に依存するようになりました。 libcurl-full を使用するシステムに PEM 10.3 パッケージをインストールする場合、--allow-erasing オプションを渡して、dnf が libcurl-minimal を削除できるようにする必要があります。	
サーバー ID ベースのフィルタリングをクラスター名前ベースのフィルタリングに置き換え、一致するプロモーションを見つけるために降格イベントの周囲に 30 秒の時間ウィンドウを追加することにより、Patroni および EFM クラスターのフェールオーバー検出ロジックを修正しました。	52030
監視対象サーバーの ASB ホストアドレスを指定する PEM 構成フラグ--asb-host-name を追加しました 監視対象サーバーの ASB のホストアドレスを明示的に指定するために新しい PEM 構成フラグ--asb-host-name が追加されました。マルチホスト文字列が ASB ホストに誤って使用された問題。	
ユーザーが選択したダウンロード形式に関係なく、チャートをダウンロードすると JPEG ファイルのみが生成される問題を修正しました。	
SQL プロファイラーから完全なトレースをダウンロードしようとしたときに発生したエラーを解決しました。	
アラートとプローブの UI 検証を修正しました。アラートとプローブ構成の管理に欠落していた UI 検証を追加し、アラート間隔が 0 分に設定されたときにフロントエンドでの警告の欠如などの問題を解決しました。	
二要素認証 2FA ページの UI の問題を修正し、入力なしで [続行] ボタンが有効になり、[キャンセル] ボタンがフォームを送信しようとしていた問題に対処しました。	

次のページに続く

表 17 – 前のページからの続き

Description	Addresses
文字列を 255 文字に制限していた pem.event_history の列タイプを変更しました。 pem.event_history テーブルの user_name、コンポーネント、操作、メッセージ、および詳細の列タイプが charactervarying(255) から TEXT に変更されて、文字列長の制限が削除され、`value toolong」エラーが解決されました。	51009
データなしで表示された円グラフに`empty data`メッセージが表示されない問題を修正しました。すべてのデータ値がゼロの場合、API は空のシリーズを返すようになりました。	
Audit、Server、および Probe Log Viewer のヘッダーにあるボタンに影響を与えるスタイルの問題を修正しました。	
`pem.server_options`ファンクションの呼び出し中に、列`connect_timeout`への誤った参照を削除しました。	52290
Barman ローソク足チャートは、進行中のバックアップたとえば、'syncing'、'started'、および'waiting for WALS' を正しく処理およびプロットするようになりました。	
Barman チャートが動的 devicePixelRatio と互換性を持つようになりました。Barman チャートのレンダリングがデバイスのピクセル比の動的な変更と互換性があるように更新され、特定のブラウザー解像度でチャートのローソク足が表示されない問題を解決しました。	
`bdr_conflict_history_summary`プローブを修正して、 `conflict_recorded`値を`bigint`として明示的にキャストし、値の小数位によって発生したプローブの失敗を解決しました。 bigint として。	
チェックが Manage Probes UI に追加され、`force_enabled`フラグが true に設定されているシステムプローブをユーザーが無効にできないようにしました。	
特にバックアップデータに進行中のステータスが含まれる場合、Barman チャートのツールチップに正しいデータが表示されない問題を修正しました。	
ルートノードを選択すると、選択が無効になっているものを含むすべての子ノードが選択される Log/Audit Manager のバグを修正しました。すべての子ノードは無効になっている場合でも。	
ジョブスケジュールマクロに条件がないため、ジョブスケジューラーの [スケジュール] タブのタイムゾーンの変更が持続しない問題を修正しました。	45933
以前に拡大されていた、ファイル>設定>その他>ユーザーインターフェイスのテーマ選択の下に表示されるプレビュー画像のサイズを修正しました。	
サーバーバージョンを正しく処理するようにログマネージャーコンポーネントを更新しました。ログマネージャーコンポーネントが更新され、EPAS / PG / PGE 18 以降のログ構成を正しくサポートおよび処理します。	
'Alert History Report' を生成しようとしたときの`string indices must be integers`問題を修正しました。	53493
パフォーマンス診断チャートの日時フォーマットの問題を修正し、x 軸の先行ゼロの省略を解決しました *04:04:00*`の代わりに`*4:4:0*	
[プロセス] タブボタンの検索ボックスと削除ボタンに影響を与える UI のズレの問題を修正しました。	

次のページに続く

表 17 – 前のページからの続き

Description	Addresses
デフォルトのメールグループのメールデータが保存されない問題を修正しました。低、高、または中のアラート通知に「デフォルトのメールグループ」が選択された場合、メール通知データが保存されなかった問題を解決しました。選択項目が送信されない場合、バックエンドはデフォルトのメールグループを正しくデフォルト設定するようになりました。	53339
Apache HTTPD の TRACE / TRACK メソッドが HTTPS 仮想ホストに対して適切に無効になっていない問題を修正しました これらのメソッドが 403 Forbidden 応答を返すように、書き換えルールが edb-ssl-pem.conf に移動されました。	53214, 54529
コンテキストメニューからダッシュボードなどを開こうとすると、サーバーまたはデータベースを右クリックして失敗するバグを修正しました。	
ダッシュボードの一部の円グラフが`Invalid literal for int() with base 10"`エラーで失敗する問題を修正しました。	
サーバー所有者とは異なるユーザーからサーバー接続パラメーターを編集するときに発生したエラーを修正し、サーバープロパティの更新と取得 API がこれらのシナリオを正しく処理できるようにします。	
unregister-agent コマンドがホストまたはポートのコンマ区切りリストを受け入れない問題を修正しました unregister-agent コマンドは、ホストまたはポートのコンマ区切りリストを正しく受け入れるように修正され、pem_port の単一の整数のみを想定していた以前の制限を解決しました。	
pemworker を使用してサーバーを登録しているときにレプリケーションソリューションを設定できない問題を修正しました。サーバーの pemworker 登録プロセスに、レプリケーションソリューションとそれに対応するオプションが正しく含まれるようになりました。	
非スーパー管理ユーザーのサーバーオプションの誤った処理が原因で、非スーパー管理ユーザーがクエリツールにアクセスしようとしたときに発生したエラーを修正しました。	
PEM の構成中に複数のホストで--host を指定すると、不適切な Web サーバーアドレスが表示される問題を修正しました 複数のホストで--host が指定された場合、configure_pem_server.shoutput の最後に不適切な Web サーバーアドレスが表示される問題を修正しました。	
pem.process_one_alert() ファンクションの SQL の問題を修正しました。pem.process_one_alert() ファンクションから pem.alert_status テーブルに行を挿入するときに列情報を明示的に提供し、アップグレード後に発生した例外 ERROR のように、列 "state_change_count"は整数型ですが、式はインターバル型です。	
サーバーが新しいサーバーグループに登録されたときにエージェントが正しくリストされないエラーを修正しました。	
接続マネージャーが、ログインしているユーザーの権限に従ってサーバー接続パラメーターを取得していない問題を修正しました。	
ユーザーの詳細が`server_options`テーブルに存在しない場合に複数のエントリを返していた`get_server.sql`を修正し、予想通りに単一のエントリが返されるようにしました。	

2.7 Postgres Enterprise Manager 10.2.0 release notes

リリース日 2025 年 8 月 14 日

Postgres Enterprise Manager 10.2 の新機能、機能強化、バグ修正、およびその他の変更には次のものが含まれます。

2.7.1 ハイライト

- フェイルオーバーイベントの新しいアラート
- レプリカの冗長アラートを自動的に無効にする
- PEM は FIPS 対応環境で実行できるようになりました
- EDB Postgres Distributed 6 モニタリングのサポート

2.7.2 機能強化

Description	Addresses
EFM / Patroni フェールオーバーイベントのアラートサポートを追加しました。PEM は、EFM および Patroni クラスターでのフェールオーバーの組み込みアラートを提供するようになりました。	
'Last Vacuum' および'Last AutoVacuum' アラートがレプリカノードで自動的に無効になるようになりました。PEM は、現在プライマリの物理レプリカであるサーバーで'Last Vacuum' および'Last AutoVacuum' アラートを自動的に無効にしません。これにより、プライマリ/スタンバイクラスターで重複したアラートを取得できません。	
EDB Postgres Distributed 6 の新しいプローブ `bdr_node_group_config_summary`および `bdr_analytic_slot_name`を追加しました。 `bdr_node_summary`プローブを更新して、`node_uuid`列を含めました。	
新しい列`parent_table_name`を`oc_table`プローブに追加して、パーティション/親テーブルの関係を示します。	
FIPS 対応環境をサポートするために、新しい構成項目 `SESSION_DIGEST_METHOD`を導入しました。新しい構成オプション `SESSION_DIGEST_METHOD`を使用すると、ユーザーは PEM Web アプリケーションでデータの署名に使用されるデフォルトの暗号化ハッシュを指定できます。デフォルトは'hashlib.sha1' です。 sha1 が非推奨であるため、将来的にこれは「hashlib.sha256」に変更されます。 `config_local.py`で `SESSION_DIGEST_METHOD = 'hashlib.sha256'`を指定することにより、今すぐ sha256 に変更できます。この変更が行われる前は、PEM は MD5 ハッシュライブラリを参照し、使用されなかった場合でも、OpenSSL が MD5 を提供しない FIPS 対応システムで PEM がクラッシュする原因となっていました。	

次のページに続く

表 18 – 前のページからの続き

Description	Addresses
モニタリングパネルの動的な名前付けを導入しました。モニタリングパネルのタブに表示されるテキストは、パネルの内容に基づいた説明的な名前になりました。これにより、複数のパネルが開いた状態での移動が簡単になります。	
タイムスタンプ値で秒を表示するように折れ線グラフツールチップを強化しました。	
PEM Windows インストーラーの HTTPD を 2.4.65 にアップグレードしました。	
自己診断クエリをシステム構成レポートに追加しました。	
チャート説明ツールチップでマークダウンフォーマットを有効にしました。	
PEM は、PEM のアップグレードまたは再構成時に、SSL ポート、SSL 証明書パス、およびサーバー名を保持するようになりました。この変更により、ユーザーは PEM のアップグレード中に上書きせずに、環境に合わせてこれらのパラメーターを変更できます。	

2.7.3 の変更

Description	Addresses
PEM エージェントから MD5 実装を削除しました MD5 ハッシュアルゴリズムを実装した PEM エージェントから未使用のコードを削除しました。	
heartbeat_interval の名前が heartbeat_tolerance に変更され、動作がわかりやすくなりました。 PEM の以前のバージョンでは、`heartbeat_interval`設定 PEM サーバーに保存されている各 PEM エージェントの設定で、PEM が待機する時間を決定しました。エージェントがダウンしていると考えられるまでのハートビート。具体的には、`2 * heartbeat_interval + 15 seconds`のハートビートがなかった場合、エージェントはダウンしていると考えられます。 15 秒は心拍間のギャップです。パラメーターの名前は、多くの場合混乱を引き起こし、ユーザーはハートビート期間を設定していると考えました。上記の式の 2 の因数も混乱を引き起こしました。 動作をより明確にするために、設定を `heartbeat_tolerance`に置き換え、式を `heartbeat_tolerance + 15 seconds`に調整しました。 PEM 10.2 にアップグレードする場合、以前の設定が尊重され、同等の `heartbeat_tolerance`に変換されます。	
Manage Probes の Manage Probes の Alternate Code タブから、11 より前の PostgreSQL および EPAS バージョンのサポートを削除しました。	
一貫性を保つために、 *unregister-server* CLI オプションを `--server-addr`の使用から `--asb-host-name`に変更しました。	

2.7.4 バグ修正

Description	Addresses
pem.server テーブルの ssl 列の欠落が原因で発生したサーバー登録の問題を解決しました PEM 10.1.1 に影響します。	
'primary' タグが PGD 固有のタグではなく PGD サーバーに表示される問題を修正しました。	
電子メールグループでコンマ区切りのメールを使用する場合の検証エラーを修正しました。	
PEM 10 の MFA に関する UI の問題に対処しました。 MFA UI が正しくレンダリングされず、したがって使用が困難または不可能である複数の問題に対処しました。	
電子メールグループで新しいエントリを追加/削除する際の既存のエントリの重複を修正しました。	
新しいメールグループを追加した後に [削除] ボタンが誤動作する問題を解決しました。	
Patroni Cluster Status プローブが非 Patroni サーバーの PEM にデータを書き込むことができる問題を修正しました。	
libpq5 が必要であるにもかかわらず、PEM 構成中にインストールされない問題を修正しました。	
Patroni サーバーでストリーミングレプリケーションダッシュボードを開くことができなかった問題を修正しました。	
リーダーが見つからない場合、または Patroni ノードリストが空の場合、正常性ステータスが誤って false に設定される Patroni DCS 正常性状態レポートの問題を修正しました。	
アラートの管理でトグルボタンと時間頻度設定を無効にする問題を修正しました	
無効な EFM インストールパスまたはクラスター名が PEM エージェントログにエラーまたは警告を生成しない問題を修正しました。	
ユーザーが Manage Probes の Manage Probes パネルを介してシステムプローブのデータ保持設定を変更できない問題を修正しました。	
フィールド名を `db_restriction`から `db_res`に更新することにより、データベース制限機能に関する問題を修正しました。	
Patroni プローブでサポートされていない機能が原因で発生した Windows ビルドの問題を修正しました	
**agent.cfg の欠落が原因で発生したエージェント登録エラーを解決しました。これにより、force-registration オプションを使用する際の デフォルト値の問題と不明なエージェントエントリが発生しました	
円グラフ設定の SQL を更新してダッシュボードレベルの設定のサポートを削除し、色の更新の問題を修正しました	
データのロード中に誤ったグラフのタイトルと円グラフの色の選択が欠落する問題を修正しました。	
棒グラフに不足している Y 軸ラベルを追加しました。	
[Alert History] ダイアログの [Download] ボタンをデフォルトで有効にしました。	

次のページに続く

表 20 – 前のページからの続き

Description	Addresses
チャート内の冗長なエラーアイコンを削除しました。	
cmtemplate テーブルグラフのしきい値関連のエラーを修正しました。	
チャートデータが空のときにエラーが表示される問題を修正しました。	
`server_versions`への冗長な API 呼び出しを削減し、代替コードテーブルのページネーションを追加しました。	
無効になった Webhook のアラート選択 UI とトグルスイッチ動作を修正しました	
チャートの色の同期の問題を修正し、チャート全体で一貫したカラーの適用を保証します。	
NGINX モードで PEM をアンインストールするときに、PEM は uWSGI を停止し、NGINX 構成をクリーンアップします。	
PEM サーバーランタイムの libjpeg 依存関係を追加しました。	
サーバーレプリケーションプロパティに基づいたストリーミングレプリケーションダッシュボードの固定グラフ	
監査構成ジョブが作成されなかった問題を修正しました。	
Capacity Manager レポートの生成に失敗していた問題を修正しました。	
カスタムプローブの間隔計算に関する問題を修正しました。	
空の文字列値を適切に処理するように Patroni タイムライン不一致アラートテンプレートクエリを更新しました。	
サーバーにストリーミングレプリケーションデータが不足している場合でも、ストリーミングレプリケーションダッシュボードを開くことができる問題を修正しました。	
ストリーミングレプリケーションダッシュボードの初期データチェック API を更新して、サーバーのレプリケーションソリューションタイプに基づいて、さまざまな監視テーブルを条件付きでチェックします。	

2.8 Postgres Enterprise Manager 10.1.2 release notes

リリース日 2025 年 7 月 21 日

これは、PEM 10 のバグ修正リリースです。

!!!重要 このリリースには、PEM サーバー 10.1.1 以降、および PEM Windows インストーラーで使用する場合の PEM エージェントすべてのプラットフォームの修正のみが含まれています。

このため、このパッチを既存の PEM サーバーインストール、または 10.1.1 より前の PEM サーバーバージョンで使用する PEM エージェントに適用する必要はありません。

Description	Addresses
`pemworker register-server`コマンドがエラーで失敗する問題を修正しました。	
Windows で PEM サーバーのインストールが失敗する問題を修正しました。	

2.9 Postgres Enterprise Manager 10.1.1 release notes

リリース日 2025 年 7 月 7 日

これは、PEM 10 のバグ修正リリースです。

2.9.1 バグ修正

Description	Addresses
パフォーマンス診断がデータを表示しない問題を修正しました。バグにより、PEM はターゲットサーバーに接続してデータを収集できません。これは、Performance Diagnostics ビューが常に空になることを意味しました。これは解決され、パフォーマンス診断が意図したとおりに動作します。	47603,48815,50207
シェル/バッチジョブが保存されない問題を修正しました。以前は、PEM は、シェル/バッチジョブを作成するときに誤ってサーバー ID をバックエンドデータベースに書き込もうとしていました。ただし、この場合サーバー ID は未定義であるため、ジョブは保存されず、エラーが表示されます。	
PEM 10.0 から PEM 10.1 へのアップグレード中に `patroni_node_status`プローブが作成されなかった問題を修正しました。	
libpq5 パッケージがインストールされる必要がある場合にインストールされない問題を修正しました。	
' 循環参照が検出されました' エラーを発生させる可能性のある v14 API の問題を修正しました。	49313
ジョブコードの一重引用符で囲まれた文字列が偽の構文エラーを発生する問題を修正しました。	
接続パラメーターがサーバープロパティに保存またはロードされない問題を修正しました。	
`send_email`ファンクションのロジックが、通常と異なる`email_group_options`に対してより堅牢になりました。	
/server API エンドポイントの PUT メソッドが失敗する問題を修正しました。この問題により、エラー `Driver.qtLiteral() missing 1 required positional argument: 'conn'`が発生しました。	
'Order denied,allow' の非推奨な指示子を Apache HTTPD 構成から削除しました。	
一部のシステムで QR コードが生成されない問題を修正しました。この問題は、各システムの前提条件 `libjpeg`ライブラリへの依存関係を追加することにより解決されました。	

2.10 Postgres Enterprise Manager 10.1.0 release notes

リリース日 2025 年 5 月 15 日

Postgres Enterprise Manager 10.1 の新機能、機能強化、バグ修正、およびその他の変更には次のものが含まれます。

2.10.1 ハイライト

- Patroni クラスターの監視とスイッチオーバーのサポート
- PGD クラスター内のノードロールの自動タグ付け
- HA PEM アーキテクチャのサポートの向上
- RHEL 9 での Kerberos 認証に関する問題を解決しました

2.10.2 機能強化

Description	Addresses
Patroni クラスターの監視とスイッチオーバーのサポートを追加しました。PEM 10.1 には、Patroni から情報を収集するための新しい'Patroni Cluster Status' が含まれています。この情報は、Patroni ノードのストリーミングレプリケーションダッシュボードに表示されます。さらに、PEM には、パトロニノード用の 5 つのサーバーレベルのアラートテンプレートが含まれるようになりました。-パトロニクラスター一時停止-パトロニ DCS が正常でない-パトロニダウンまたは切断中-パトロニリーダーが検出されません-パトロニタイムラインの不一致	
PEM は、PGD ノードにロールを自動的にタグ付けするようになりました。新しいプローブ「PGD Node Roles」が PEM に追加されました。このプローブは、各 PGD ノードのロールを収集します。プローブによって収集されたデータは、ロールごとにノードに自動的にラベルを付けるために使用されます。	
Web アプリケーションから PEM サーバーへのマルチホスト接続文字列のサポートを追加しました PEM Web アプリケーションを構成するときに、PEM バックエンドに複数のホストアドレスを指定できるようになりました。 Web アプリケーションは、書き込み接続を受け入れるものが見つかるまで、それぞれを順番に試行します。これは、 *Platform Compatibility* と組み合わせて使用する必要があります。	
マルチホスト接続文字列 PEM エージェントのサポートを PEM バックエンドデータベースサーバーに追加しました。PEM エージェントを構成するときに、PEM バックエンドに複数のホストアドレスを指定できるようになりました。エージェントは、書き込み接続を受け入れるものが見つかるまで、それぞれを順番に試行します。これは、 *Platform Compatibility* と組み合わせて使用する必要があります。	

次のページに続く

表 23 – 前のページからの続き

Description	Addresses
`pem.register_pem_server(server_id)`ファンクションを導入して、データベースサーバーをレプリカ PEM サーバーとして登録できるようにしました高可用性 PEM アーキテクチャを構成する場合、このオプションを使用して PEM バックエンドの追加コピーを登録できます。このファンクションは、これらすべてのサーバーの静的システムタスクを作成しますが、デフォルトでは無効になっています。これらはプライマリでのみ有効になります。昇格後、新しいプライマリ PEM サーバーは、システムタスクを自動的に引き継ぎます。これにより、PEM の以前のバージョンに存在したシステムジョブを実行するサーバー/エージェント ID 1 の依存関係が削除されます。	
新しい API バージョン v14 を追加しました	
スケジュールされたタスクの詳細ビュー内に`Steps`サブパネルを導入しました。このパネルには、タスクに関連付けられているすべてのステップと、各ステップの最後の実行ステータスと実行時間が表示され、可視性とモニタリングが向上します。	
エージェント/サーバーを登録するときにクラスター名を指定するために pem-worker の`--cluster-name`スイッチを追加しました。	
PEM エージェントをクラスターに含めることができるようになりました	
PEM は、ジョブの完了/失敗時に Webhook を呼び出すことができるようになりました。Webhook エンドポイントを構成する場合。 'Alert' または'Job' Webhook タイプを選択できるようになりました。 'job' タイプの Webhook は、各ジョブの通知設定に従って起動されます。	
無効なプローブに依存するチャートから有効にする機能を追加しました。	

2.10.3 の変更

Description	Addresses
pgAdmin4 9.2 をマージしました。 PEM 10.1 は pgAdmin 9.2 ベースであるため、このリリースから fixes and features の大部分を継承します。	
データが欠落している場合など、チャートに情報メッセージを表示する PEM の機能を追加しました。	
pywinpty パッケージを Windows の依存関係に追加しました。	
アラートによって作成されたジョブは、PEM オブジェクトエクスプローラーにリストされなくなります。アラートによって作成されたジョブは、以前はオブジェクトエクスプローラーに表示されていました。これにより、ユーザーはこれらのジョブを意図しない方法で編集でき、通常は破損が発生するため、混乱が発生しました。	

2.10.4 バグ修正

Description	Addresses
ダッシュボードのエラーにより、その後にアクセスされるすべてのダッシュボードが正しくロードされない問題を修正しました。	
RHEL 9 でカーベロスを使用して PEM の認証中に psycopg エラーが発生する問題を修正しました PEM は RHEL 9 の Kerberos と完全な互換性があるため、この組み合わせのユーザーは PEM 10.1 にアップグレードすることをお勧めします。	
ダッシュボードが編集後に更新されない問題を修正しました。	
プローブコードに特殊文字を含むカスタムプローブが適切に作成されない問題を修正しました。	47765
'stop Trace' が予想通りに機能しなかった SQL プロファイラーの問題を修正しました。	
値を変更するときにプローブ構成が UI から保存されなかった問題を修正しました。	
JSON データパーサー例外が原因で EFM プローブが失敗する問題を修正しました。	46527
アラートステータステーブルの日付から 1970 年が得られる問題を修正しました。	
ユーザーにエラー `Column "Cluster Name" does not exist`が表示されるストリーミングレプリケーションチャートの問題を修正しました。	47332
PEM 構成中にエラー `The error - module 'lib' has no attribute 'X509_V_FLAG_NOTIFY_POLICY'`が発生する可能性のある問題を修正しました。	
しきい値以外のアラートフィールドの更新時に「threshold」キーエラーが発生していた問題を修正しました。	
チャートが変更後に自動的に更新されない問題を修正しました。	
チャートの凡例にアクセスできない問題を修正しました。	
EPAS インスタンスが見つからない場合、Audit Manager が有用なメッセージを表示しない問題を修正しました。	
PEM から EFM スイッチオーバーを実行するときにブラウザーで発生する可能性のあるエラーを修正しました。	
新しいテンプレートを追加した後、[アラートテンプレート] ページが適切に表示されない問題を修正しました。	
クラスタータイプノードでオブジェクトを選択した後、アラートが Manage Alerts パネルから消えていた問題を修正しました。	48476
列名の特殊文字が Capacity Manager チャートで予期しない動作を発生させた問題を修正しました。	
ポインターをチャートの外に移動した後、ツールチップが消えない問題を修正しました。	
ラインチャートの render_as_query ファンクションにエラーキャッチロジックがありません。	
`agent_ssl_passphrase_script`スクリプトを実行する際の例外処理が向上しました。	

次のページに続く

表 25 – 前のページからの続き

Description	Addresses
PieChart の NaN 値を修正しました。	
デフォルトグループで削除ボタンを無効にする必要がある場合に有効になるメールグループの問題を修正しました。	
ブラウザコンソールログに「duplicate key」エラーを発生させるバグを修正しました。	
応答を待機している間の Webhook の'test' ボタンの状態に関する問題を修正しました。	
UMASK が 0027 の場合、NGINX / uWSGI での PEM 構成障害を修正しました	
ipv6 が無効になっているときに NGINX を起動できない問題を修正しました	
execute_alert_on がトグルされたときにアラートが保存されない問題を修正しました。	
ダッシュボード編集後のリダイレクトに関する問題を修正しました。	
コンテキストメニューからダッシュボードが開かれたときに、ブレッドクラムツリーが適切に表示されない問題を修正しました。	
[Register/Edit server] ダイアログから欠落しているチームフィールドを修正しました。サーバーの登録時に特権チェックも追加しました。	

2.11 Postgres Enterprise Manager 10.0.0 release notes

リリース日 2025 年 3 月 24 日

注釈

RHEL 同様のシステムで PEM 10 にアップグレードする場合、古い edb-pem-docs パッケージの削除によってアップグレードがブロックされないように、次の dnf コマンドを使用します。

```
dnf upgrade edb-pem --allowerasing
dnf autoremove
```

2.11.1 ハイライト

- サーバーロールの自動タグ付けを使用した、サーバーとエージェントのクラスターへのグループ化のサポート
- タブをよりよく整理するためのワークスペースを含む、更新されたよりパフォーマンスの高いユーザーインターフェイス
- 有効期限が切れたデータの最適化された削除によるパフォーマンスの向上
- NGINX Web サーバーのサポート

2.11.2 機能強化

Description	Addresses
オブジェクトエクスプローラーでのクラスターのサポートを追加しました。PEM 10 では、クラスターの概念を導入しています。これらは、高可用性 HA クラスター、たとえば、Postgres Distributed または EDB Failover Manager を使用するクラスターを表すために使用されるサーバーとエージェントのグループです。オブジェクトエクスプローラーでクラスターを作成および設定できます。将来のリリースでは、この機能を拡張して、PEM での HA クラスターを操作するエクスペリエンスを向上させます。	
サーバーとエージェントのタグ付けのサポートを追加しました。PEM は、サーバーの任意のタグ付けをサポートするようになりました。現在、これは単に表面的な機能です。将来的には、タグによるフィルタリングと選択をサポートするように拡張する予定です。	
UI を React フレームワークに移植し、多数の使いやすさの向上をもたらしました。 PEM ユーザーインターフェイス全体が React フレームワークに移植されました。これにより、PEM プロジェクトが pgAdmin と連携し、上流からの機能強化が簡単になります。また、より現代的でパフォーマンスの高いインターフェイスも提供します。この移植を利用して、UI に多数の改善を行いました。-プローブ/アラートにページネーションを追加しました。-ワークスペースを実装しました。-スケジュールされたタスクをタスクと履歴タブに分割しました。-アラート、プローブ、その他を作成するためのダイアログビューを実装しましたオブジェクト、グリッドビューを置き換えます。	
pgAdmin からの最近の機能強化が組み込まれています。PEM 10 の更新されたアーキテクチャにより、pgAdmin から以下を含む多くの機能強化をプルできます。- エンティティ関係図の生成- クエリツール結果のグラフのプロット- systemstats 拡張機能を使用してリアルタイムのシステムメトリックダッシュボードを表示	
IO 分析プローブ用に `read_operations`および `write_operations`列を追加しました。このデータは、クラウドストレージを使用する場合に特に関連する IOPS 秒ごとの入力/出力操作に関する監視およびアラートに使用できます。	
server_info **`プローブに** *`node_type* **`列を追加しました。** *`node_type* **`は、次の値のいずれかを取ります。** *`primary*、*replica*、*readonly*、*standalone*。	
管理が簡単になるように PEM タブをワークスペースに分離しました。次の PEM コンポーネントのパネル/タブが'Enterprise Manager' ワークスペースで開きます。-アラートテンプレート-電子メールグループ-Webhook-カスタムプローブ-チャートの管理次の PEM コンポーネントのパネル/タブは、'Enterprise Tools' ワークスペースで開かれます。- SQL プロファイラー-パフォーマンス診断-キャパシティマネージャーレポート次の PEM コンポーネントのパネル/タブが'Object Explorer' ワークスペースで開きます。-プローブ構成-アラート	

次のページに続く

表 26 – 前のページからの続き

Description	Addresses
デフォルトの Web サーバーとして NGINX のサポートを追加しました。PEM 10 は、Linux の Web サーバーとして Apache HTTPD と NGINX の両方をサポートしています。 Windows では、Apache HTTPD のみがサポートされています。新規インストールのデフォルトは NGINX です。 `/usr/edb/pem/bin/switch-web-server.sh`スクリプトを実行して Web サーバーを切り替えることを選択しない限り、PEM 9 からのアップグレードは Apache HTTPD を使用し続けます。	
SQL プロファイルに「最小クエリ期間」のサポートを追加しました。SQLSQL でトレースを作成するときに、最小クエリ期間を指定できるようになりました。これにより、長時間実行されるクエリのみをトレースできます。つまり、サイズ制限を超えずに、非常に長い期間トレースを実行できることを意味します。	
Ubuntu 24.04 のサポートを追加しました。	
PEM エージェントは、誤ったアラートを回避するために PEM サーバーに接続した後、しばらく待機するようになりました。これは、 `agent.cfg`の`alert_reconnect_delay`パラメーターを介して構成できます。	
EFM 5.0 のサポートを追加しました。 'xlog ' **列の名前は** *'lsn* ' に変更されます。	
PEM は、サーバーがプライマリかレプリカかを示すタグを自動的に追加します。	
グローバルアラートテンプレート「サーバーダウン」および「エージェントダウン」の詳細情報 SQL を改善しました。	
パージ履歴機能のパフォーマンスが向上しました。	

2.11.3 の変更

Description	Addresses
Log Analysis Expert、Tuning Wizard、Postgres Expert、および Index Advisor が削除されます。これらの機能は PEM 10.0 以降から削除されました。 Postgres チューニングの場合、EDB Postgres Tuner 拡張機能と accompany web utility をお勧めします。 EDB Server のインデックスアドバイザー拡張機能は非推奨であり、すべての Postgres ディストリビューションと互換性のある EDB Query Advisor に置き換えられました。ログ分析には、 pgBadger をお勧めします。	
PEM バックエンドデータベースは自動的に調整されなくなりました。 EDB Postgres Tuner を使用して PEM バックエンドデータベースを調整することをお勧めします。	
マウントポイントが io_ana Analysis プローブから欠落している問題を修正しました。	
オペレーティングシステムのダッシュボードに影響を与えるバグを修正し、ネットワークトラフィック情報が表示されなかった。	40938

2.11.4 バグ修正

Description	Addresses
`pem_manage_dashboard`を持つユーザーがカスタムダッシュボードを表示できない問題を修正しました。	
マウントポイントが io_ana Analysis プローブから欠落している問題を修正しました。	
オペレーティングシステムのダッシュボードに影響を与えるバグを修正し、ネットワークトラフィック情報が表示されなかった。	40938

第3章　Platform compatibility

PEM がサポートするプラットフォームとバージョンの詳細については、Platform Compatibility を参照してください。

> **注釈**
>
> Postgres Enterprise Manager 8.3 以降は SLES でサポートされています。

> **警告**
>
> RHEL 9.6 の互換性を保証するには、PEM 10.3 ARM または 10.3.1 Intel/ARM を使用しているユーザーは PEM 10 にアップグレードする必要があります

第4章　Hardware requirements for installing the PEM server on Linux

サーバーを監視し、ダッシュボードをレンダリングする際の最適なパフォーマンスを得るには、少なくとも次のシステムに PEM をインストールすることをお勧めします。

- 4 つの CPU コア
- 8 GB の RAM
- 100 GB のストレージ

データストレージには追加のディスク領域が必要です。リソースの使用量は、定義および有効化されたプローブと、監視対象データベースのアクティビティレベルによって異なります。PEM を使用しながらサーバーリソースを監視すると、初期システム構成を拡張する必要がある場合に通知されます。

第5章　PLANNING

5.1 Understanding Postgres Enterprise Manager components and architecture

Postgres Enterprise ManagerPEM は、単一のグラフィカルインターフェイスを介して複数の Postgres サーバーを監視および管理します。PEM は、インフラストラクチャの次の領域を監視できます。

- **ホスト** - 1 つ以上のサーバー物理または仮想とそのオペレーティングシステム。
- **データベースサーバー** - ホストで実行されている PostgreSQL、 EDB Postgres Advanced Server、または EDB Postgres Extended Server 旧 2ndQPostgres の 1 つ以上のインスタンス。
- **データベース** - 1 つ以上のデータベースと、テーブルやインデックスなどのスキーマオブジェクト。

> **注釈**
>
> Postgres という用語は、 PostgreSQL、 EDB Postgres Advanced Server、または EDB Postgres Extended Server を指します。

PEM は個々のソフトウェアコンポーネントで構成されています。

- **PEM サーバー** - PEM サーバーは、データを監視するためのデータリポジトリであり、エージェントとクライアントが接続するサーバーです。PEM サーバーは、PostgreSQL のインスタンス、監視データを保存するための関連データベース、および Web サービスを提供するサーバーで構成されています。
- **PEM エージェント** - PEM エージェントは、タスクを実行し、エージェントホストおよび監視対象の Postgres インスタンスから PEM サーバーに統計を報告します。単一の PEM エージェントは、エージェントがインストールされている同じホストまたは複数のリモートホストに存在する Postgres の複数のインストール済みインスタンスを監視できます。PEM サーバーインストールの一部としてエージェントをインストールするか、スタンドアロンエージェントをインストールできます。
- **PEM Web クライアント** - PEM Web インターフェイスを使用すると、Postgres サーバーを管理および監視し、PEM 拡張機能を使用できます。Web インターフェイスソフトウェアは、PEM サーバーと共にインストールされ、サポートされている Web ブラウザーを使用してアクセスされます。
- **SQL プロファイラー** - SQL プロファイラーは、PEM で分析する SQL プロファイラーツールの監視データとクエリプランを記録する Postgres サーバー拡張機能です。これは PEM のオプショナルコンポーネントですが、拡張機能は、使用する Postgres の各インスタンスで有効にする必要があります。SQL プロファイラーは、PEM サーバーで管理されているものだけでなく、Postgres の EDB ディストリビューションのサポートされているバージョンで使用できます。詳細およびサポートされているバージョンについては、*Installing the SQL Profiler extension* を参照してください。

5.1.1 PEM アーキテクチャ

次のアーキテクチャ図は、PEM サーバー、クライアント、管理対象および非管理対象の Postgres サーバー間の関係を示しています。

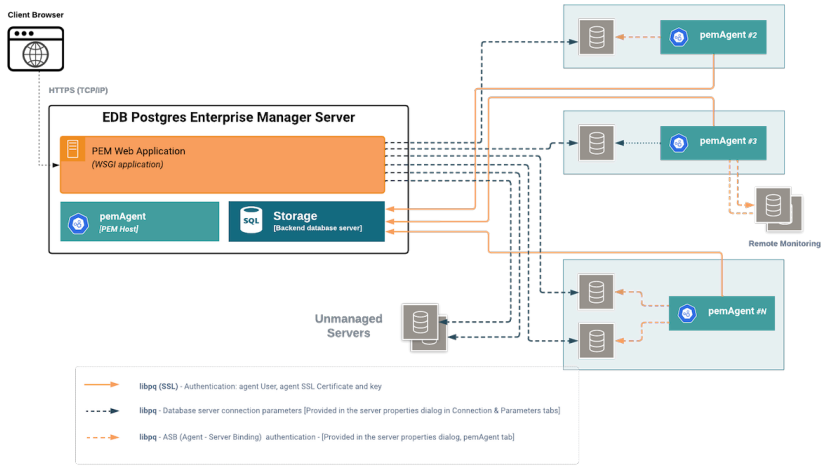


図 1: PEM Architecture

5.1.1.1 PEM サーバー

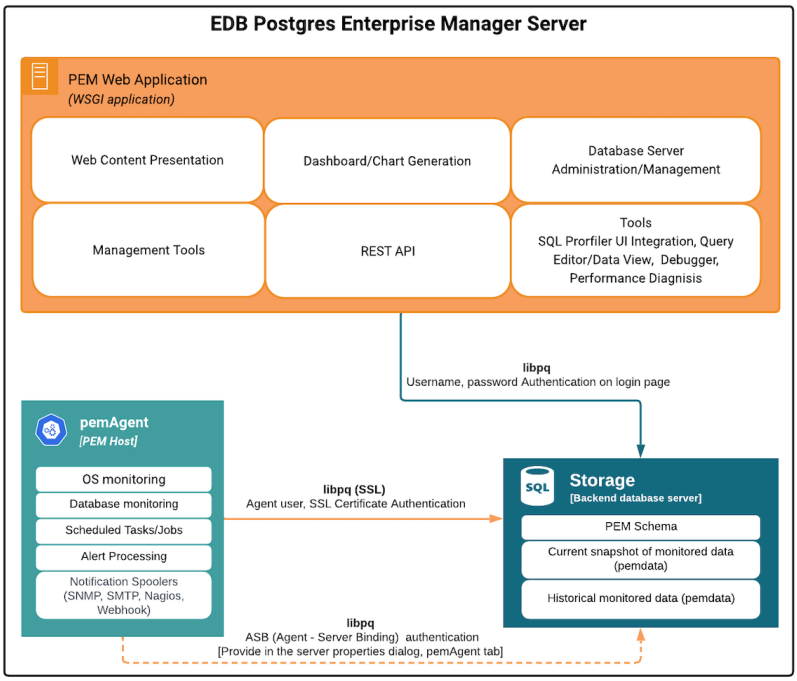


図 2: PEM Server

PEM サーバは Postgres のインスタンス、クライアントに Web サービスを提供する Web サーバ、および PEM エージェントで構成されています。PEM は、サーバーサイドの暗号化プラグインを使用して認証証明書を生成します。

Postgres のインスタンスと Web サーバーは、同じホスト上または別のホスト上にあってもかまいません。

> **注釈**
>
> すべての PEM 機能は、選択したバックエンドデータベースサーバー PostgreSQL または EDB Postgres Advanced Server で利用できます。

- **Postgres インスタンスデータベースサーバー** - これはバックエンドデータベースサーバーです。 `pem` という名前のデータベースをホストし、PEM サーバーのリポジトリとして機能します。`pem` データベースには、各監視対象ホスト、サーバー、およびデータベースから収集されたメトリックデータを保存するいくつかのスキーマが含まれています。
- `pem` - このスキーマは PEM アプリケーションのコアです。アプリケーションに必要な構成ファンクション、テーブル、またはビューの定義が含まれています。
- `pemdata` -このスキーマは、監視対象データの現在のスナップショットを保存します。
- `pemhistory` -このスキーマは、履歴監視対象データを保存します。
- **Web サーバー Nginx/HTTPD** - PEM Web アプリケーションは、Web サーバーの背後に WSGI アプリケーションとして展開され、クライアントに Web サービスを提供します。これは次で構成されています。
- **Web コンテンツプレゼンテーション** - プレゼンテーションレイヤーは、Web アプリケーションブラウザやログインページなどによって作成されます。
- **Rest API** - REST API により、他のコンテンツと統合できます。アプリとサービス.
- **データベースサーバの管理/management** - 管理対象および非管理対象サーバーの CREATE、ALTER、DROP などのデータベースサーバの管理と管理アクティビティを実行できます。
- **ダッシュボード/チャートの生成** - 内部的に、Web アプリケーションには、ダッシュボードとチャートを生成する機能が含まれています。
- **管理ツール** - Audit Manager、Capacity Manager、および Log Manager は Web アプリケーションで使用できます。
- 他のツールは、管理対象または非管理対象サーバーで機能を提供します。
- **SQL プロファイラー UI の統合** - SQL プロファイラーは、セッションコンテンツの簡単に分析されたトレースを生成します。
- **クエリエディタ/データビュー** - クエリエディタでは、データを照会、編集、および表示できます。edb_tran_20 **Debugger** - デバッガーは、クエリーのデバッグに役立ちます。
- **パフォーマンス診断** - パフォーマンス診断は、Postgres インスタンスのパフォーマンスの分析に役立ちます。

専用マシンを使用して、PEM バックエンドデータベースの運用インスタンスをホストすることをお勧めします。監視対象のデータベースサーバーの数、およびサーバーが処理しているワークロードによっては、ホストは高レベルのデータスループットの影響を受ける場合があります。

5.1.1.2 PEM エージェント

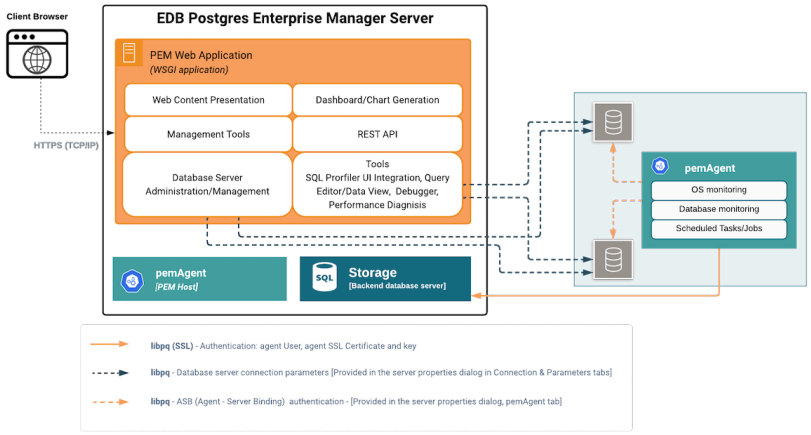


図 3: PEM Agent

PEM エージェントは、マシンとオペレーティングシステムと、それらがバインドされている各 Postgres インスタンスから監視データを収集します。各 PEM エージェントは 1 つの物理マシンまたは仮想マシンを監視でき、他のシステムにインストールされている複数のデータベースサーバーをローカルに監視できます。これらのサーバーは、同じシステムまたはリモートにインストールできます。また、サーバーのシャットダウン、SQL プロファイラーのトレース、カスタムジョブなど、ユーザーがスケジュールした他のタスクを実行する責任もあります。

PEM エージェントは、PEM サーバーのインストール時にデフォルトで PEM サーバーにインストールされます。通常、PEM ホスト上の PEM エージェントと呼ばれます。別に、PEM を使用して監視する Postgres インスタンスをホスティングする他のサーバーに PEM エージェントをインストールすることもできます。

ローカルで監視するかリモートでモニタリングかにかかわらず、PEM エージェントは、SSL 証明書ベースの認証を使用して、PostgreSQL の libpq を使用して PEM サーバーに接続します。Windows の PEM エージェントインストーラーと Linux の pemworker CLI は、各エージェントを PEM サーバーに登録し、必要な証明書を生成およびインストールします。

PEM エージェントと PEM サーバーの間は一方通行だけです。PEM エージェントは常に PEM サーバーに接続します。

PEM エージェントは、監視対象の各データベースサーバーに接続できる必要があります。この接続は、TCP/IP 接続またはオプションで Unix ホスト上の Unix ドメインソケットを介して確立され、オプションで SSL を使用できます。監視対象サーバーへの接続と認証を構成する必要があります。

構成が完了すると、各エージェントは、ホストと、監視対象の各データベースサーバーとデータベースに関する統計情報やその他の情報を収集します。各情報は_metric_と呼ばれ、_probe_によって収集されます。ほとんどのプローブは、効率のために一度にマルチプルのメトリックを収集します。収集されるメトリックの例は次のとおりです。

- ディスク I/O 統計
- ネットワーク統計
- データベースサーバーバージョン文字列
- データベースサーバー構成オプション GUC 値

- テーブルアクセス統計
- テーブルとインデックスのサイズ

PEM プローブのリストについては、 プローブ を参照してください。

デフォルトでは、データベースサーバにバインドされた PEM エージェントは OS /データベースモニタリング統計を収集し、その特定のデータベースサーバのスケジュールされたタスク/ジョブを実行し、PEM サーバーの pem データベースにデータを保存します。

アラート処理、SNMP / SMTP スプーラー、および Nagios スプーラーデータは、PEM サーバー上の pem データベースに保存され、デフォルトで PEM ホスト上の PEM エージェントによって処理されます。ただし、PEM エージェントの SNMP / SMTP および Nagios パラメーターを調整することにより、他の PEM エージェントによる処理を有効にできます。

これらのパラメーターの詳細については、 サーバー構成パラメーター - リファレンス を参照してください。

5.1.1.3 PEM Web アプリケーション

Web アプリケーションは PEM サーバーに接続し、管理対象または非管理対象サーバーと、それらに存在するデータベースとスキーマを直接管理できます。

Web アプリケーションでは、ダッシュボードや Capacity Manager などの機能を介してサーバーに記録されたデータを利用する PEM 機能を使用できます。

5.1.1.4 SQL プロファイラー拡張機能

すべてのサーバーに SQL プロファイラー拡張機能をインストールする必要はありませんが、SQL プロファイラーを使用する各サーバーに拡張機能をインストールして有効にする必要があります。監視されていない開発サーバーに SQL プロファイラー拡張機能をインストールして有効にすることもできます。アドホックに使用するために SQL プロファイラー拡張機能を一時的にインストールすることもできます。

PEM で管理されていないサーバーで SQL プロファイラーを使用できます。ただし、スケジュールされたトレースを実行するには、サーバーに拡張機能をインストールして有効にし、インストールおよび構成済みの PEM エージェントで管理する必要があります。

詳細については、 *Installing the SQL Profiler extension* および *Using SQL Profiler* を参照してください。

5.2 Deployment considerations

Postgres Enterprise Manager を展開する前に、次の要素を検討してください。

Considerations	Implementation instructions
PEM を実行するにはどのような仕様のサーバーが必要ですか?	*Sizing PEM deployments*
私の EDB サブスクリプションは、PEM の使用計画をカバーしていますか?	*Platform Compatibility*

次のページに続く

表 1 – 前のページからの続き

Considerations	Implementation instructions
スタンドアロンサーバーで十分ですか、それとも高可用性アーキテクチャが必要ですか?	Installing the server or *Platform Compatibility*
接続プーリングを実装する必要がありますか?	*Platform Compatibility*
SSL 証明書の生成と更新はどのように管理すればよいですか?	*Deciding on an SSL model*
どのような種類のユーザー認証を使用すればよいですか?	*Authentication options*
セキュリティの脆弱性を回避するにはどのようなアクションをとればよいですか?	*Securing your deployment*
Web サーバーはどこでホストすればよいですか?	*Platform Compatibility*

5.2.1 How PEM licensing works

PEM は、EDB でサポートされている Postgres インスタンスのモニタリングと管理に使用することを目的としています。そのため、PEM を使用する資格のあるサブスクリプションがある場合、スタックに追加してこれらのインスタンスを監視しても、通常、追加のライセンスコストは発生しません。

5.2.1.1 ライセンスルール

PEM を使用するには、EDB Standard または Enterprise サブスクリプションが必要です。

PEM を使用して監視する Postgres サーバーは、EDB Standard または Enterprise サブスクリプションの対象である必要があります。これは、PEM エージェントをローカルにインストールする場合、リモートモニタリングを介して、または PEM Web アプリケーションからクライアント接続を行う場合に当てはまります。

コアを消費しない非 Postgres サーバーたとえば、PGD プロキシノードまたは Barman を実行しているマシンなどは、PEM によって監視される場合があります。PEM でこれらのサーバーを監視することにより、追加のコアは消費されません。

PEM バックエンドとして使用される Postgres インスタンスは、サブスクリプションからコアを消費しません。同様に、他の PEM コンポーネントによってコアは消費されません。これらのコンポーネントはすべて、サポート契約の対象です。

5.2.1.2 例

完全にサポートされた環境への PEM の追加

32 コアの Enterprise サブスクリプションを契約している顧客が、EDB Postgres Advanced Server を実行している 8 × 4 コアサーバーがあるため、32 コアを完全に消費すると仮定します。この顧客は、5 番目のサーバーに PEM をインストールし、それを使用して 8 つの EDB Postgres Advanced Server サーバーを監視できます。PEM サーバーはコアを消費せず、監視対象の EDB Postgres Advanced Server インスタンスは既に完全にカバーされているため、この構成では、サブスクリプションを変更する必要はありません。

同様に、この顧客が Barman サーバーを追加し、モニタリングのために PEM に接続した場合、追加のコアは消費されません。

サポートされていないサーバーの PEM への追加

36 コアの Standard サブスクリプションをお持ちのお客様に、EDB サポートの対象となっている PostgreSQL を実行している PEM および 6 x 6 コアサーバーがあり、前の例のように 36 コアを消費すると仮定します。

顧客はまた、EDB サブスクリプションの対象になっていない 10 個の 2 コア PostgreSQL サーバーを所有しています。これらのサーバーは、Standard または Enterprise サブスクリプションの対象になっていないため、PEM によって監視されません。顧客がこれらのサーバーを監視する場合は、サブスクリプションにさらに 20 コアを追加する必要があります。

> **注釈**
>
> 追加の 20 コアの EDB Community 360 サブスクリプションでは、十分ではありません。 PEM によって監視されるサーバーは、Standard または Enterprise でカバーされている必要があります。

5.2.2 PEM server and agent connection management mechanism

適度な数を介して、または Postgres Enterprise ManagerPEM の接続プーリングを使用して、接続を効果的に管理できます。

非 Windows マシンで PEM エージェントから PEM サーバーへの接続数を制限するための接続プーラーとして PgBouncer を使用できます。

以下では、PEM エージェントが接続プーラーとして PgBouncer を使用する場合と使用しない場合の両方でデータベース接続を管理する方法を確認できます。

5.2.2.1 PgBouncer なし

デフォルト構成では、各 PEM エージェントは、暗号化に SSL を使用して PEM データベースサーバーに直接接続します。各 PEM エージェントは、接続に独自の専用ユーザーを使用します。

たとえば、ID 1 のユーザエージェントは、agent1 として PEM データベースサーバーに接続します。

5.2.2.2 PgBouncer を使用する場合

PgBouncer 対応環境では、PEM エージェントは PEM データベースサーバーに直接接続できません。PEM エージェントは、PgBouncer への接続専用に構成したプロキシユーザーを使用する必要があります。この例では、すべての PEM エージェント接続を処理するプロキシユーザーの名前は `pem_agent_user1` です。

PEM エージェントが SSL を使用して PgBouncer に接続すると、PgBouncer は PEM データベースサーバーへの接続要求を管理します。PgBouncer は、接続プーリング用に構成されたルールを使用して、たとえば、アクティブな接続の確立された最大数を尊重することにより、着信接続要求を管理します。

!!!note 接続プーラーとしては PgBouncer バージョン 1.9.0 以降を使用します。バージョン 1.9.0 以降は、証明書認証をサポートしています。

PEM を使用した PgBouncer の構成の詳細については、*Connection pooling using PgBouncer* を参照してください。

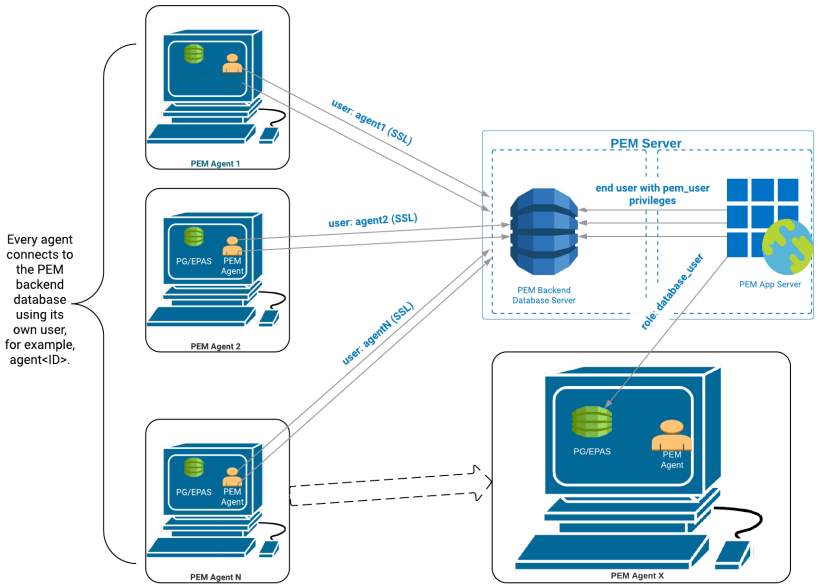


図 4: Connecting to the PEM database without PgBouncer

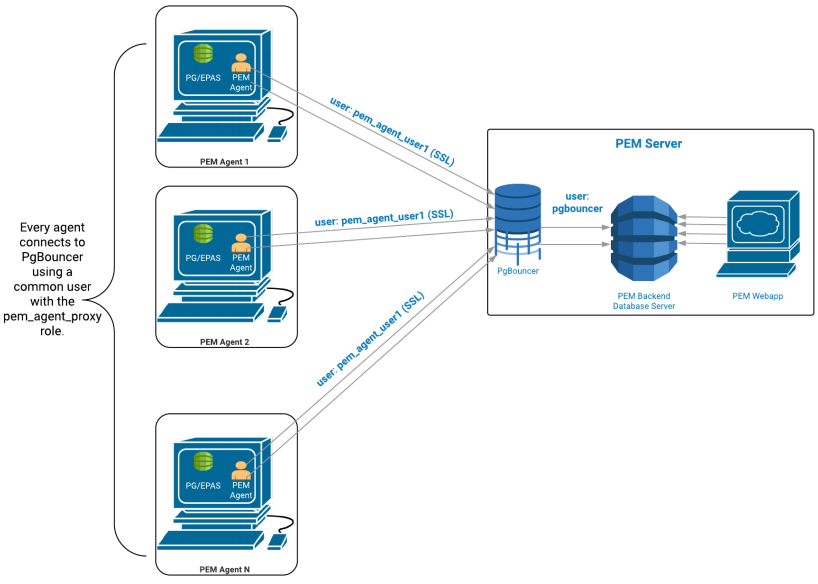


図 5: Connecting to pgBouncer.

5.2.3 Authentication options

PEM ユーザー認証は Postgres バックエンドサーバーに委任されます。特定のユーザー名で PEM にログインできるようにするには、バックエンドにその名前の Postgres ユーザーが存在する必要があります。Web アプリケーションホストから `pem` データベースに接続するには、ユーザーが `pem_user` ロールを持ち、`pg_hba.conf` によって許可されている必要があります。PEM ユーザーロールの詳細については、*PEM 認証の管理* を参照してください。

デフォルトでは、PEM はユーザーにパスワードを使用した認証を必要とします。オプションで 2FA を適用したり、Kerberos を使用したパスワードレス認証を有効にしたりできます。

5.2.3.1 1. 二要素認証 2FA

PEM は 2FA をサポートして、標準のログイン資格情報を超えた追加のセキュリティレイヤーを提供します。`config_local.py` ファイルのパラメーターを変更することにより構成できます。

- **サポートされている方法**
- **電子メール** - ワンタイム確認コードをユーザーのメールアドレスに送信します。
- **Authenticator App** - Google Authenticator のような TOTP ベースのアプリケーションを使用します。
- **キー構成**
- **MFA_ENABLED** - 機能を有効にするには `true` に設定します。
- **MFA_FORCE_REGISTRATION** - `true` の場合、次回ログイン時に 2FA に登録する必要があります。
- **メールサーバー**

電子メールベースの 2FA の場合、PEM はその内部 SMTP 構成または Flask-Mail を使用できます。

- **ドキュメント**

Configuring two-factor authentication 。

5.2.3.2 2.Kerberos 認証

Kerberos は、SSO 環境に安全なチケットベースの認証メカニズムを提供します。この構成は、Kerberos サーバー、PEM サーバー、およびクライアントマシン全体のホスト名と DNS 解決に大きく依存しています。

Linux ベースのケルベロス

この方法には、Kerberos 認証サーバーのインストールと、PEM Web アプリケーション HTTP とバックエンドデータベース Postgres の両方のプリンシパルの構成が含まれます。

- **キータブ** - セキュリティキータブは Kerberos サーバーから抽出され、PEM サーバーに展開されて、暗号化された通信が促進されます。
- **認証** - ユーザーは `kinit` ユーティリティを介してチケットを取得します。pg_hba.conf ファイルは、gss Generic Security Service 方法を使用するように更新されました。
- **ドキュメント** - *Configuring Kerberos authentication on Linux* 。

Windows Active Directory SSPI

PEM は、Kerberos 認証のために Windows Active Directory ドメインサービスと統合できます。

- **サービスマッピング** - Active Directory ユーザーは、特に HTTP および Postgres サービスプリンシパルに作成され、マッピングされます。
- **キータブ生成** - Windows `ktpass` ユーティリティは、Windows ドメインと対話する Linux ベースの PEM サーバーのキータブを生成するために使用されます。
- **ブラウザ要件** - クライアントブラウザ Firefox/Chrome は、SPNEGO / Kerberos ネゴシエーションのために PEM サーバー URI を信頼するように構成する必要があります。
- **ドキュメント** - *Configuring kerberos authentication on Windows* 。

5.2.4 Deciding on an SSL model

PEM エージェントは、*PEM アーキテクチャ* で詳細に説明しているように、libpq を使用して PEM バックエンドデータベースに直接接続する Postgres クライアントです。

通信を安全に管理および暗号化するために、PEM は Postgres バックエンドのネイティブ SSL/TLS 機能を活用します。

このセキュリティモデルが機能するには、次のコンポーネントを構成する必要があります。

- **エージェント認証**すべての PEM エージェントには、割り当てられた Postgres ユーザーの名前と一致するクライアント証明書が必要です。
- **ルート信頼**バックエンド Postgres サーバーをルート CA ファイルで構成して、受信エージェント証明書を検証する必要があります。
- **サーバー暗号化**バックエンド Postgres サーバーには、データストリームを暗号化し、ID を確立するために独自の SSL 証明書とプライベートキーが必要です。

ニーズとポリシーに応じて、これらの証明書とキーを提供するために使用できるさまざまなモデルがあります。以下の情報は、インストールを開始する前にどのモデルに従うかを決定するのに役立ちます。

5.2.4.1 証明書管理モデル

自動 sslutils 経由 または **手動外部 CA またはローカル CA を使用**の 2 つの方法でセキュリティ証明書を処理するように PEM を構成できます。適切なパスの選択は、インフラストラクチャの複雑さとコンプライアンスのニーズによって異なります。

- **sslutils で自動化** sslutils 拡張機能を活用することにより、PEM は独自の認証局 CA として機能します。エージェントが登録すると、PEM は必要なクライアント証明書を自動的に生成します。

以下の用途に最適 速度と利便性が優先される標準の非 HA 展開。

欠点 厳密な外部 CA ポリシーを使用する組織の制御が低下します。

長所 pgBouncer と唯一の互換性のある方法トランザクションはプールされる同じユーザーからである必要があるため。少数の証明書の管理がはるかに簡単になります。

短所特定のエージェント接続を監査するのが困難です。証明書を失効すると、すべてのエージェントが切断されます。

2. **一意のエージェントユーザー個人の証明書**すべてのエージェントには、独自の一意の証明書と、対応する番号の Postgres ユーザーたとえば、agent1、agent2 などがあります。

長所 高い監査可能性。1 つの証明書を失効すると、1 つのエージェントにのみ影響します。

短所 大量の証明書による多大な管理オーバーヘッド。

- **ローカル CA 手動** sslutils に依存する代わりに、PEM ホストまたは他の場所でローカル CA を手動で構成し、それを使用して PEM サーバーとエージェントの証明書を発行できます。上記の **外部 CA** の場合と同じアーキテクチャの選択があります。これは、sslutils を使用できず pgBouncer を使用する予定がある場合、外部/組織 CA がない場合にのみお勧めします。

5.2.5 Securing your deployment

攻撃に対して PEM 展開を強化するには、次の対策を検討してください。

- PEM、オペレーティングシステム、およびサードパーティのライブラリが定期的に更新されていることを確認します。最新のセキュリティパッチが適用されていない場合、システムはサイバー攻撃に対して脆弱です。PEM が使用するシステムパッケージの詳細については、*Dependencies of the PEM Server and Agent on Linux* を参照してください。
- PEM サーバーとして使用される Postgres インスタンスが最新の状態に保たれていることを確認し、Postgres security best practices を適用します。
-

Secure the web server 。

- *security settings of the PEM web application* を必要に応じて構成します。

5.2.5.1 Web server security configuration

PEM Web サーバーのセキュリティを構成できます。

Windows では、サポートされている Web サーバーは Apache HTTPD です。 Apache HTTPD は、PEM HTTPD という名前で PEM にバンドルされています。 Apache HTTPD 構成ファイルは `pem.conf` であり、SSL 構成ファイルは `httpd-ssl-pem.conf` です。どちらの構成ファイルも`<Apache_Installation_Path>/conf/addons` ディレクトリにあります。

Linux では、NGINX と Apache HTTPD の両方がサポートされています。NGINX 構成ファイルは、RHEL 様のシステムでは`/etc/nginx/conf.d/edb-pem.conf` であり、Debian 様のシステムでは`/etc/nginx/sites-available/edb-pem.conf` です。 Apache HTTPD 構成ファイルは `edb-pem.conf` であり、SSL 構成ファイルは `edb-ssl-pem.conf` です。どちらの構成ファイルも`<Apache_Installation_Path>/conf.d` ディレクトリにあります。

デフォルトで適用される推奨事項

これらの推奨事項は、PEM の新規インストールにデフォルトで適用されます。Web サーバー構成をカスタマイズした場合、またはかなり古いバージョンの PEM から引き継いだ場合、この情報を使用して、構成が現在の標準を満たしていることを確認できます。

安全でない SSL および TLS プロトコルを無効にする

PEM の新規インストールでは、SSL バージョン SSLv2、SSLv3、TLS 1、および TLS 1.1 はデフォルトで無効になっています。これらのバージョンは最も脆弱であり、暗号化に関する懸念があります。

NGINX の場合、PEM は構成ファイルに次の行を追加します。

```
ssl_protocols TLSv1.2 TLSv1.3;
```

Apache HTTPD の場合、PEM は次の行を SSL 構成ファイルに追加します。

```
SSLProtocol -All TLSv1.2
```

```
SSLProxyProtocol -All TLSv1.2
```

次のコマンドを使用して、TLS 1.1 が無効になっていることを確認できます。URL を Web サーバーの URL に置き換えます。戻り値 35 は、TLS 1.1 が無効であることを意味します。 0 は有効になっていることを意味します。

```
curl -k -v -s --tls-max 1.1 https://pem-server:8443 >/dev/null 2>&1; echo $?
```

Web サーバーの情報漏えいを無効にする

PEM の新規インストールでは、サーバートークンとサーバー署名を無効にすることにより、クライアントに公開されるサーバーに関する情報を最小限に抑えるように Web サーバーが構成されます。サーバートークンは、応答ヘッダーでサーバーに関する情報を公開します。サーバー署名は、エラーメッセージなどのサーバー生成ページのフッターの情報を公開します。

NGINX の場合、PEM は構成ファイルに次の行を追加します。

```
server_tokens off;
```

Apache HTTPD の場合、PEM は次の行を SSL 構成ファイルに追加します。

```
ServerTokens Prod
ServerSignature Off
```

ディレクトリリストを無効にする

ディレクトリリストにより、攻撃者は、コンテンツが提供されるディレクトリの完全なコンテンツを表示できます。このリストにより、攻撃者はアプリケーションをリバースエンジニアリングして、ソースコードを取得し、考えられるセキュリティ上の欠陥が分析され、アプリケーションに関する詳細情報を発見する可能性があります。

このリスクを回避するために、PEM はディレクトリリストを無効にします。

NGINX の場合、PEM は `autoindex:  off` を設定します。

Apache HTTPD の場合、PEM は `Options -Indexes` 指示子を設定します。

```
<Directory /application/directory> Options -Indexes </Directory>
```

クロスサイトトレース

TRACE および TRACK HTTP メソッドは、サーバーのデバッグに使用されます。 HTTP TRACE 要求がサポートされている Web サーバーに送信されると、サーバーは応答して、HTTP ヘッダーを含む自分に渡されたデータをエコーします。Apache 構成でこれらのメソッドを無効にすることをお勧めします。

NGINX では、 TRACK および TRACE メソッドはデフォルトで無効になっています。Apache HTTPD では、PEM は構成ファイルに次の行を含めて、これらのメソッドを拒否します。一部のスキャナーはこの構文を理解せず、これらのメソッドが許可されていると誤って報告する場合があります。

```
RewriteEngine on
RewriteCond %{REQUEST_METHOD} ^(TRACE|TRACK|OPTIONS)
RewriteRule .\* - [F]
```

次のコマンドで、TRACK および TRACE が無効になっていることを確認できます。URL を Web サーバーの URL に置き換えます。戻り値 35 は、TLS 1.1 が無効であることを意味します。0 は有効になっていることを意味します。メソッドが無効になっている場合、コマンドはテキスト `405 Method Not Allowed` または同様のものを含む HTML 応答を結果ます。

```
curl -kL -X TRACK https://pem-server:8443/pem
curl -kL -X TRACE https://pem-server:8443/pem
```

セキュリティのために HTTP ヘッダーを最適化する

PEM は、セキュリティを向上させるためにさまざまな HTTP ヘッダーオプションを設定します。これらの設定は、`config.py` および `config_distro.py` ファイルで定義されます。このファイルは、Linux では`/usr/edb/pem/web` 、 Windows では `C:\ProgramFiles\edb\pem\server\share\web` にあります。

これらの設定を変更する場合は、これらのファイルを編集しないでください。代わりに、同じ場所に `config_local.py` という名前のファイルを作成または既に存在する場合は編集し、目的の設定を追加します。これらの設定は、`config.py` および `config_distro.py` ファイルの設定をオーバーライドします。これらは PEM のアップグレード中に上書きされません。

`config.py` ファイルの詳細については、*Managing configuration settings* を参照してください。

X フレームオプション

X-Frame-Options は、ブラウザーが<iframe>タグでページをレンダリングできるかどうかを示します。特にクリックジャッキングから保護します。PEM には、これらの攻撃を防ぐためのホスト検証 `X_FRAME_OPTIONS` オプションがあり、 `config_local.py` ファイルで構成できます。デフォルトは次のとおりです。

```
X_FRAME_OPTIONS = "SAMEORIGIN"
```

コンテンツセキュリティポリシー

Content-Security-Policy は、HTML5 標準の一部です。置き換える X-Frame-Options ヘッダーよりも幅広い保護を提供します。ウェブサイトの作成者がドメインをホワイトリストに登録できるように設計されています。作成者は、ホワイトリスト登録されたドメイン、およびページを埋め込むことができるドメインからスクリプト、スタイルシート、フォントなどのリソースをロードできます。

PEM には、攻撃を防止するためのホスト検証 `CONTENT_SECURITY_POLICY` オプションがあり、`config_local.py` ファイルで構成できます。デフォルトは次のとおりです。

```
CONTENT_SECURITY_POLICY = "default-src https: data: blob: unsafe-inline 'unsafe-eval;
↪"
```

厳密なトランスポートセキュリティ

Strict-Transport-Security HSTS 応答ヘッダーは、中間者攻撃を防止できます。このオプションを有効にすると、Web サイトまたは Web アプリケーションは、HTTPS のみを受け入れ、HTTP は受け入れないことをブラウザーに指示します。デフォルトは次のとおりです。

```
STRICT_TRANSPORT_SECURITY = "max-age=31536000;includeSubDomains"
```

> **注釈**
>
> このパラメーターを追加すると、構成が変更された場合に問題が発生する可能性があります。したがって、PEM のインストールが完了してテストされた後にのみ追加することをお勧めします。

X-Content-Type-Options

X-Content-Type-Options 応答 HTTP ヘッダーはマーカーです。サーバーはマーカーを使用して、Content-Type ヘッダーでアドバタイズされた MIME タイプを変更および追跡できないことを示します。次は、MIME タイプスニッフィングをオプトアウトする方法、つまり、MIME タイプが意図的に構成されていることを示します。デフォルトは次のとおりです。

```
X_CONTENT_TYPE_OPTIONS = "nosniff"
```

X-XSS-プロテクション

クロスサイトスクリプティング XSS は、Web サーバーで最も一般的なアプリケーションレイヤーの脆弱性の 1 つです。XSS を使用すると、攻撃者は他のユーザーが表示する Web ページにクライアントサイドスクリプトを挿入できます。ヘッダーへの HTTP X-XSS-Protection 応答は、Internet Explorer、Chrome、および Safari の機能です。反射されたクロスサイトスクリプティング XSS 攻撃を検出すると、ページのロードを停止します。これらの保護は、サイトがインライン JavaScript 'unsafe-inline' の使用を無効にする強力なコンテンツセキュリティポリシーを実装している場合、最新のブラウザーでは必要ありません。ただし、これらの保護は、CSP をまだサポートしていない古い Web ブラウザーのユーザーに保護を提供できます。デフォルトは次のとおりです。

```
X_XSS_PROTECTION = "1; mode=block"
```

Cookie のセキュリティ

Cookie は、構成データを保存するためにサーバーがブラウザーに送信するデータの小さなパケットです。ブラウザーはそれらと他のすべての要求を同じサーバーに送信するため、Cookie を保護する方法を知ることが重要です。`config.py` の複数の構成オプションにより、Cookie を安全にできます。これらは 3 つの最も重要なオプションです。

- SESSION_COOKIE_SECURE - このフラグは、暗号化されていない接続を介して Cookie が送信されないようにします。ブラウザーは、暗号化されたチャネルがないとサーバーへの要求に Cookie を追加できません。ブラウザーは、HTTPS などの接続にのみ Cookie を追加できます。デフォルトは次のとおりです。

```
SESSION_COOKIE_SECURE = True
```

- SESSION_COOKIE_HTTPONLY - デフォルトでは、JavaScript は Cookie のコンテンツを読み取ることができます。`HTTPOnly` フラグは、スクリプトが Cookie を読み取ることを防ぎます。代わりに、ブラウザーは HTTP または HTTPS 要求でのみ Cookie を使用します。ハッカーは XSS の脆弱性を悪用して Cookie の内容を学習することはできません。たとえば、`sessionId` Cookie は、クライアントサイドスクリプトで読み取られる必要はありません。したがって、`sessionId` Cookie の `HTTPOnly` フラグを設定できます。デフォルトは次のとおりです。

```
SESSION_COOKIE_HTTPONLY = True
```

- ENHANCED_COOKIE_PROTECTION - このオプションを `True` に設定すると、IP アドレスとユーザーエージェントに基づいてトークンが生成されます。後続のすべての要求で、トークンは再計算され、最初の要求で計算されたものと比較されます。セッション Cookie が盗まれ、攻撃者が別の場所から使用した場合、生成されるトークンは異なります。その場合、拡張機能はセッションをクリアし、要求をブロックします。デフォルトは次のとおりです。

```
ENHANCED_COOKIE_PROTECTION = True
```

> **注釈**
>
> このオプションは、Kubernetes またはロードバランサーの背後などの動的 IP アドレスホスティング環境にサーバーが展開される場合、問題を発生させる可能性があります。これらの場合、このオプションを `False` に設定します。

変更を適用するには、Web サーバーを再起動します。

`config.py` ファイルの詳細については、*Managing configuration settings* を参照してください。

手動で適用できる追加の推奨事項

これらの推奨事項は、PEM が展開されている環境に固有の追加情報またはアクションを必要とするため、自動的に適用されません。

SSL 証明書を使用したセキュア HTTPD

PEM 構成中に、自己署名証明書が生成され、Web サーバーとクライアント間のトラフィックを保護します。セキュリティを強化し、サイトが安全でないというブラウザーの警告を防ぐには、replace this certificate with one signed by a trusted certificate authority をお勧めします。

特権のないユーザアカウントから Web サーバーを実行する

Linux では、PEM は OS が提供する Web サーバーパッケージを使用します。通常、これらはルートユーザーとして Web サーバーを実行するサービスユニットを作成します。

root ユーザーとして Web サーバーを実行すると、セキュリティ問題が発生する可能性があります。一意の非特権ユーザーとして Web サーバーを実行することをお勧めします。これにより、セキュリティ侵害中に実行されている他のサービスの保護に役立ちます。

Windows では、 `WSGIDaemonProcess` 指示子と機能が利用できないため、Web サーバーと WSGI アプリの両方がシステムユーザー `LocalSystem` アカウントとして実行されます。

Web サーバーへのアクセスを、ビジネスニーズと互換性のある最小の IP アドレスのセットに制限することをお勧めします。これは、ファイアウォール構成などを介してネットワークインフラストラクチャレベルで最も一般的に実行されますが、Web サーバーによって適用されることもあります。PEM アプリケーション構成ファイル`<PEM_INSTALLATION_PATH>/web/config_local.py` は、この目的に `ALLOWED_HOSTS` 構成パラメーターをサポートしています。

例

```
#  You can specify one or more subnets:

ALLOWED_HOSTS = [225.0.0.0/8, 226.0.0.0/7, 228.0.0.0/6]

#  You can specify individual host addresses:

ALLOWED_HOSTS = [127.0.0.1, 192.168.0.1]
```

アプリケーション構成ファイルの変更を適用するには、Web サーバーを再起動します。

5.2.5.2 PEM application security configurations

セッションタイムアウト

Web アプリケーションでセッションの有効期限を長すぎると、他のセッションベースの攻撃の危険が増加します。攻撃者は、有効なセッション ID を再利用し、関連するセッションをハイジャックする時間が増えます。セッション間隔が短いほど、攻撃者が有効なセッション ID を使用する時間が短くなります。このセキュリティ問題を回避するには、Web アプリケーションの非アクティビティタイムアウトを低い値に設定することをお勧めします。

PEM では、ユーザーセッションのタイムアウト値を設定できます。Web コンソールで指定された期間にユーザーアクティビティがない場合、PEM はユーザーを Web コンソールからログアウトします。PEM 管理者は、非アクティブな時間を設定できます。この値は、各ユーザーではなく、アプリケーション全体に対するものです。

タイムアウト期間を構成するには、 <PEM_INSTALLATION_PATH>/web ディレクトリの config_local.py ファイルの USER_INACTIVITY_TIMEOUT パラメーターを変更します。デフォルトでは、このパラメーターは無効になっています。値を秒単位で指定します。

たとえば、アプリケーションが 15 分間非アクティブになった後にユーザーをログアウトするように指定するには

```
USER_INACTIVITY_TIMEOUT = 900
```

変更を適用するには、Apache サービスを再起動します。

config.py ファイルの詳細については、*Managing configuration settings* を参照してください。

RestAPI ヘッダーのカスタマイズ

要件を満たすように RestAPI トークンヘッダーをカスタマイズできます。デフォルト値は config.py ファイルでは公開されません。 config_local.py ファイルで、次のヘッダーをカスタマイズします。

PEM_HEADER_SUBJECT_TOKEN_KEY

この構成オプションを使用すると、HTTP ヘッダー名を変更して、生成されたトークンを取得できます。デフォルトでは、トークンを作成する要求を送信すると、サーバー応答に X-Subject-Token ヘッダーが含まれます。このヘッダーには、新しく生成されたトークンの値が含まれています。ヘッダー名をカスタマイズする場合は、config_local.py ファイルを更新できます。

```
PEM_HEADER_SUBJECT_TOKEN_KEY = Pem-RestAPI-Generate-Token
```

このコマンドは、次の出力を生成します。

```
curl -ik -X POST -d {"username":"enterprisedb","password":"edb"} -H "Content-Type:␣
↪application/json" https://localhost:8443/pem/api/token/
HTTP/1.1 201 CREATED
Date: Thu, 29 Oct 2020 11:03:48 GMT
Server: Apache
Content-Length: 326
Pem-RestAPI-Generate-Token: 997aef95-d46d-4d84-932a-a80146eaf84f
```

PEM_HEADER_TOKEN_KEY

この構成オプションを使用すると、HTTP 要求のヘッダー名を変更できます。このヘッダー名を使用して、トークンを PEM サーバーに送信できます。デフォルトでは、トークンを生成する要求を送信するときのトークンヘッダー名は X-Auth-Token です。RestAPI 要求ヘッダー名をカスタマイズする場合は、config_local.py ファイルを更新できます。

```
PEM_HEADER_TOKEN_KEY = Pem-Token
```

この設定により、トークンを送信できます。

```
$ curl -Lk -X GET -H "Pem-Token: gw5rzaloxydp91ttd1c97w24b5sv60clic24sxy9" https://
↪localhost:8443/pem/api/v4/agent
```

PEM_TOKEN_EXPIRY

この構成オプションを使用すると、PEM RestAPI トークンの有効期限を生成後に変更できます。デフォルトでは、トークンの有効期限は 20 分 1200 秒に設定されています。たとえば、トークンの有効期限を 10 分に変更するには、次のように config_local.py ファイルを更新します。

```
PEM_TOKEN_EXPIRY = 600
```

変更を適用するには、Apache サービスを再起動します。

PEM のロールベースのアクセス制御

ロールベースのアクセス制御 RBAC は、組織内のユーザーのロールに基づいてアプリケーションへのアクセスを制限します。これは、アクセス制御の主要な方法の 1 つです。RBAC のロールは、ユーザーのアプリケーションへのアクセスレベルを参照します。ユーザーは、ジョブを実行するために必要な情報のみにアクセスできます。PEM のロールは継承可能で、添字的ではなく追加的です。つまり、PEM 管理者は、最下位レベルのロールをユーザーに付与し、ユーザーがジョブを実行するために必要なロールを付与する必要があります。例、SQL プロファイラーへのアクセスのみを付与するには

```
CREATE ROLE user_sql_profiler WITH LOGIN NOSUPERUSER NOCREATEDB NOCREATEROLE INHERIT␣
↪NOREPLICATION CONNECTION LIMIT -1 PASSWORD xxxxxx;
GRANT pem_user, pem_comp_sqlprofiler TO user_sql_profiler;
```

ロールの詳細については、 *PEM roles* を参照してください。

パスワード管理

PEM 管理ユーザーのためのセキュリティのヒントの 1 つは、PEM ログインパスワードを新しいものに定期的に変更することです。パスワードの変更

- 複数のアカウントの侵害を防止します
- 常時アクセスを防止
- 物理的に保護されていないシステムでの保存されたパスワードの使用を防止します
- キーストロークロガーによって取得されるアクセスを制限します

非 root ユーザーで PEM エージェントジョブを実行する

ほとんどの場合、PEM エージェントは root ユーザーとしてインストールされ、root 権限を持つデーモンプロセスとして実行されます。デフォルトでは、PEM はスケジュールされたジョブ/タスクの実行を無効にします。PEM は、PEM エージェント構成ファイルを変更することにより、非 root ユーザーとしてスケジュールされたジョブを実行するサポートを提供します。

スケジュールされたジョブを非 root ユーザーとして実行するには、`agent.cfg` ファイルの `batch_script_user` パラメーターのエントリーを変更し、スクリプトを実行するユーザーを指定します。非 root ユーザーまたは root ユーザー ID を指定できます。ユーザーを指定しない場合、または指定され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。非 root ユーザーが PEM エージェントを実行している場合、`batch_script_user` の値は無視されます。PEM エージェントの実行に使用されるのと同じ非 root ユーザーがスクリプトを実行します。

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次の場所にあります。

```
HKEY_LOCAL_MACHINE\Software\EnterpriseDB\PEM\agent
```

PEM エージェントと PEM バックエンドデータベースサーバー証明書の変更

デフォルトでは、PEM をインストールすると、インストーラーは PEM エージェントと PEM データベースサーバーの自己署名証明書を生成および使用します。PEM エージェントは、PEM データベースサーバーに接続するときにこれらの証明書を使用します。PEM エージェントと PEM データベースサーバーに独自の SSL 証明書を使用するには、Managing certificates を参照してください。

> **注釈**
>
> PEM は、カスタムの場所への SSL CA 証明書の配置をサポートしていません。 *ca_certificate.crt* および `ca_key.key` の場所は変更しないでください。

5.2.6 Web server installation options

PEM サーバーのインストール中に、Web サーバーのホスティング設定を指定できます。実稼働環境の場合、ベストプラクティスは PEM サーバーと Web サーバーを別のホストに配置することです。

5.2.6.1 別のホスト上の PEM サーバーと Web サーバー

1. 両方のホストに PEM サーバーをインストールします。 *Installing the PEM server and PEM-HTTPD on separate hosts* を参照してください。
2. 最初のホストで **データベース**オプションを選択して、PEM サーバーホストを構成します。
3. 2 番目のホストで **Web Services** オプションを選択して、Web サーバーを構成します。

PEM サーバーの構成の詳細については、 *Configuring the PEM server on Linux* を参照してください。

5.2.6.2 同じホスト上の PEM サーバーと Web サーバー

1. PEM サーバーをインストールします。 *Installing the PEM server and PEM-HTTPD on separate hosts* を参照してください。

2. 構成スクリプトを実行します。PEM サーバーと Web サーバーを同じホストにインストールするには、 **Web Services and Database** オプションを選択します。*Configuring the PEM server on Linux* を参照してください。

5.2.7 Sizing PEM deployments

適切な量のリソースを PEM サーバーに割り当て、PEM のリソース要件を促進する重要な要素を理解することが重要です。T シャツのサイズは、開始するのに役立ちます。

!!!重要 PEM は Postgres です PEM は基本的に Postgres データベースに基づいて構築されており、コアロジックのほとんどは SQL ファンクションを介して実行されます。PEM を標準の Postgres インスタンスとして扱うことが重要です。最適な応答性を保証には、従来の Postgres の構成と調整のベストプラクティスをすべて遵守してください。PEM はパフォーマンスも監視します。ダッシュボード、 *Performance Diagnostic* 、*Upgrading SQL Profiler* などの統合ツールを使用して、その動作を分析および理解できます。

5.2.7.1 リソース要件に影響を与える要因

エージェント/接続の数

エージェントの数は、Postgres データベースへの接続数を直接決定します。すべての接続が新しいプロセスを開始するため、接続数は Postgres 展開全体のリソース消費における重要な要素です。

デフォルトでは、ほとんどのエージェントは 1 つの接続を使用します。`enable_heartbeat_connection` が true に設定されている場合、各エージェントは 2 つの接続を使用します。`alert_threads` 、`enable_smtp` 、`enable_webhooks` 、または `enable_snmp` セットで構成されたエージェントは、これらの機能用に追加の接続を開きます。

これらの接続の多くは、ほとんどの時間アイドル状態であるため、`max_connections` を他の方法よりもわずかに高く設定しても安全です。ただし、これはまだ Postgres であることに注意することが重要です。接続数が 100 を超えると、接続オーバーヘッドが急速にリソースを消費します。このため、大規模な PEM 展開の場合、pgBouncer を展開して接続を効率的に管理することを強くお勧めします。

適度な数または適切な接続プーリングを使用して接続を効果的に管理すると仮定すると、メモリと CPU のニーズは次の要素によって決定されます。

頻繁にアクセスされるデータのサイズ - メモリ

メモリサイズの推奨事項は、最も頻繁に使用されるデータが Postgres の共有バッファに完全に存在し、ディスク I/O を最小限に抑え、パフォーマンスを最大化するように設計されています。この優先度の高い、頻繁にアクセスされるデータには次のものが含まれます。

- `pemdata` スキーマ、すべてのプローブによって収集された最新のデータポイントを保存します。
- プローブの実行とアラートディスパッチを管理するために使用される `pem` スキーマのさまざまなテーブル。

この重要なデータのサイズは、主にプローブの合計数と各プローブが返すデータの量によって決まります。実際、このサイズは、監視対象データベースオブジェクトの数、特にテーブルやインデックスのような多数のオブジェクトによって決まります。

以下の推奨サイズは、RAM の約 25%が共有バッファー専用にされるという仮定に基づいています。

プローブとアラートの実行速度 - CPU およびアラートスレッド

PEM は、新しいプローブデータの取り込みと、構成されたアラートしきい値と比較してそのデータの評価という 2 つの重要なタスクを常に実行します。したがって、PEM には、設定済みのすべてのプローブとアラートを指定された周波数で実行するための十分な CPU 時間が必要です。たとえば、1 分に 1 回実行するように設定された 100 のアラートを使用したセットアップでは、1 分あたり 100 のアラート評価の処理能力が必要です。この必要な実行レートを満たさないと、アラートが遅延されるか、アラートが完全にトリガーされない可能性があります。

アラートとプローブの実行数は次のようにスケールします。

- プローブとアラートの数
- プローブとアラートの設定された頻度

推奨される CPU 数は、PEM がデフォルト構成を使用してすべてのプローブとアラートに必要な実行レートを維持できるように計算されます。多くの追加アラートまたはプローブを有効にするか、実行周波数を増やすことによりセットアップをカスタマイズする場合、遅延を防ぐためにそれに応じて CPU をスケーリングする必要がある場合があります。

!!!重要 アラートスレッド大規模な展開では、PEM ホスト上の PEM エージェント構成のアラートスレッドの合計数を増やす必要があります。アラートスレッドの数は、PEM がアラート評価に使用できる CPU コア数を制限するため、これを正しく調整してアラート遅延の発生を回避し、PEM にプロビジョニングされた CPU リソースを効果的に利用することが重要です。

PEM ホストで実行されている PEM エージェントの `alert_threads` 設定を変更することにより、アラートスレッドの数を変更できます。

アラート評価が複数のスレッドで共有される効率は PEM 10.5 で大幅に向上したため、既存のシステムでこのパラメーターを調整する前に、PEM バージョンが少なくとも 10.5 であることを確認してください。

PEM のストレージニーズは、主に保持する履歴データの量によって決まります。

この量を促進する 3 つの主な要因。

- 監視対象オブジェクト数 - 収集されたデータの生の量通常はテーブルやインデックスなどの多数のオブジェクトによって占められます。
- データの更新頻度 - プローブデータが変更される頻度 PEM は圧縮を適用するため、繰り返される未変更の値が追加の領域を消費しません。
- データ保持期間 - 履歴データが保持される設定された期間。

推奨されるストレージサイズは、PEM のデフォルトのプローブ頻度、標準の保持期間、および推定データ圧縮率に基づいています。

5.2.7.2 PEM T シャツのサイズ

重要

上記で説明したすべての理由から、PEM サーバーの負荷は、監視している不動産の性質に大きく依存します。サイズガイドは開始点です。大規模なサイズの場合、監視対象サーバーのサブセットを PEM に追加し、リソース使用量を測定することから始めることを強くお勧めします。最終的なリソースニーズが私たちが提案するものと大幅に異なっていても、驚かないでください。

CPU およびアラートスレッドの要件は、エージェントの数よりも速くスケールするため、システムのサイズを 2 倍にすると、リソース要件が単に 2 倍になると考えないことにも注意してください。

edb_excla_notran_3 .. csv-table::

header
Size,Description,Component,CPUs,RAM,Storage,IOPS,Other Notes

widths
5,10,5,5,5,5,5,8

align
left

class
longtable

Nano,PEM のテスト用、1〜2 の監視対象サーバー、シングルユーザー,—,1,2 GB,20 GB,<100,—Tiny,少数の Web アプリユーザーを含む単一の HA クラスターの場合,—,2,4 GB,50 GB,<100,—Small,"≤ 50 サーバー、5 人未満の同時 Web ユーザーの場合。 >50,000 のテーブル/インデックスの場合、より高くなります",—,3,6 GB,200 GB,300,—Medium,"サーバー 100 以下の場合、最大 5 人の同時 Web ユーザー。 >100,000 のテーブル/インデックスの場合、より高くなります",—,4,8 GB,400 GB,600,Use **pgBouncer** if heartbeat connections are enabled Large,"サーバー 300 以下の場合、同時ユーザー数〜10。 >250,000 のテーブル/インデックスの場合、より高くなります",Backend,6,16 GB,2 TB,2000,"~2 alert threads, **pgBouncer** " "",NA,Frontend,1–2,2–4 GB,20 GB,<100,—X Large,"サーバー 600 以下の場合、同時ユーザー 10 人以上。 >500,000 テーブル/インデックスの場合、より高くなります",Backend,10,32 GB,3 TB,4000,"~5 alert threads, **pgBouncer** " "",NA,Frontend,2,4 GB,20 GB,100,—XX Large,"サーバー ≤ 1000 の場合、同時ユーザー 10 人以上。 >1000,000 のテーブル/インデックスの場合、より高くなります",Backend,22,48 GB,4 TB,6000,"~12 alert threads, **pgBouncer** " "",NA,Frontend,2,4 GB,20 GB,100,—

適切な調整と十分なリソースがあれば、1000 を超えるエージェントを含む PEM システムを実行できます。ただし、追加される各エージェントの増分リソースコストの増加は、大規模な PEM システムが複数の小規模な PEM システムよりもはるかに効率が低いことを意味します。このため、単一のシステムが 600〜800 エージェントを超えないように、PEM システム全体に不動産をセグメント化することをお勧めします。

5.2.8 High availability patterns for PEM deployment

PEM 展開の可用性を高くする必要がある場合は、高可用性 HA トポロジで PEM バックエンドデータベースとフロントエンド Web アプリケーションの複数のインスタンスを展開できます。

一部の一般的な構成のインストール手順の詳細については、*Installing Postgres Enterprise Manager in HA Patterns* を参照してください。

5.2.8.1 PEM バックエンドデータベースの複数のインスタンスの実行

高可用性 PEM バックエンドデータベースは、他の標準の HA Postgres クラスターと同様に機能します。バックエンドの主要な要件は次のとおりです。

プライマリ

任意の時点で、PEM バックエンドデータベースの 1 つのインスタンスのみが書き込み接続を受け入れることができます。このインスタンスは、*primary PEM* **バックエンドデータベース**、または単に_primary_と呼ばれます。

他のすべてのインスタンスは、プライマリのレプリカとして機能する必要があります。プライマリ障害が発生した場合にレプリカを昇格させるための信頼できる方法を実装する必要があります。これには通常、サポートされているフェールオーバーマネージャーの使用が含まれます。

PEM は、次のフェイルオーバーソリューションをサポートしています。

- EDB フェールオーバーマネージャー EFM
- パトローニ

> **注釈**
>
> EDB Postgres Distributed は、PEM バックエンドとしてサポートされていません。

接続はプライマリに移動する必要があります

アクティブな PEM Web アプリケーションインスタンスと実行中の PEM エージェントからのすべての接続は、プライマリ PEM バックエンドデータベースに接続する必要があります。

これを保証には 2 つの一般的なアプローチがあります。

1. 常にプライマリにルーティングする *C1:* **単一のエンドポイントとコロケーション** を使用します。これは、仮想 IPVIP、ネットワークロードバランサー、または現在のプライマリにトラフィックをルーティングするプロキシなどの方法を使用して実装できます。
2. プライマリが変更されたときにエンドポイントをスイッチするようにクライアントを構成します。これは一般的にマルチホスト接続文字列を使用して実行され、プライマリが到達不能な場合にクライアントは代替ホストに対して再試行できます。

詳細については、 **ロードバランサー、プロキシ、および** *VIP* および **リファレンスアーキテクチャ** を参照してください。

5.2.8.2 複数の PEM Web アプリケーションインスタンスの実行

前述のように、Web アプリケーションの複数のインスタンスを実行し、すべてプライマリ PEM バックエンドに接続することにより、PEM Web アプリケーションの高可用性を実現できます。バックエンドとは異なり、複数のフロントエンドインスタンスは問題なく同時に実行できます。

すべてのインスタンスで一貫した動作を保証するには

- ユーザー設定をローカルではなく PEM バックエンドデータベースに保存するようにすべてのインスタンスを構成します。
- ダンプ/リストアなどのファイルを生成または保存する機能の場合、すべての Web インスタンスにアクセスできる共有ファイルシステムを使用します。

5.2.8.3 HA PEM 展開のアップグレード

HA 構成で PEM を実行する場合、明確に定義されテストされたアップグレード手順を行うことが重要です。これにより、ダウンタイムが最小限に抑えられ、アップグレードプロセス中にバックエンドデータベースと Web アプリケーションの両方の整合性が維持されます。HA PEM をアップグレードする手順については、*Upgrading an HA PEM installation* を参照してください。特定の環境でこれらの手順を検証します。

5.2.8.4 ロードバランサー、プロキシ、および VIP

リファレンスアーキテクチャでは、プロキシ/VIP という用語は、インバウンド接続の単一のエンドポイントを提示し、トラフィックを現在のプライマリにルーティングするコンポーネントを指します。これは、さまざまなツールとテクニックを使用して実装できます。ここで概要を説明します。

フェールオーバーマネージャーによって管理される仮想 IPVIP を使用する

仮想 IPVIP は、特定の物理ネットワークインターフェイスに関連付けられない IP アドレスであり、サブネット内のノード間を移動できます。VIP は OS ネットワークスタックによって管理されるため、追加のハードウェアまたはサードパーティソフトウェアは必要ありません。

VIP を使用してトラフィックをプライマリにルーティングするには、フェールオーバー中に VIP を新しいプライマリに割り当てるようにフェールオーバーマネージャーを構成します。

- EDB Failover ManagerEFM は、VIP をネイティブにサポートします。この方法を選択する場合、EFM の使用をお勧めします。
- 制限 VIP は、マルチリージョンまたはマルチサブネットのクラウド環境には適していません。

ロードバランサーの使用

ご使用の環境がロードバランサーたとえば、AWS Elastic Load Balancer、オンプレミス F5 などをサポートしている場合、それらを使用してトラフィックを現在のプライマリにルーティングできます。一般的なパターンは 2 つあります。

- フェールオーバー中にロードバランサーを再構成するフェールオーバーマネージャーを使用して、インバウンド接続を新しいプライマリにルーティングします。

- ロードバランサーが HTTP エンドポイントを介して各ノードをポーリングし、現在のプライマリが判断するフェールオーバーマネージャーを使用します。プライマリのみが肯定的な応答たとえば、HTTP コード 200、ロードバランサーが 1 つのポーリング間隔以内にすべてのトラフィックをプライマリにスイッチすることを意味します。これは、たとえば、Patroni の`/primary` または`/read-write` エンドポイントを介して実現できます。

2 番目のパターンは、ロードバランサーが動作するように設計されており、フェールオーバー中にロードバランサーの構成を変更するための管理アクセスを必要としないため、一般的に好まれます。

注釈

「ご使用の環境がロードバランサーを提供する場合」と言う理由は、このソリューションを適切に実装して、単一障害点になることを回避する必要があるためです。ロードバランサーは可用性が高い必要があります。これには、通常、DNS フェールオーバー、Elastic IP、またはその他のインフラストラクチャレベルのソリューションを使用して、単一障害点になることを回避することが含まれます。

データベースの前に HAProxy または pgBouncer のスタンドアロンインスタンスを追加するだけでは十分ではありません。EDB サポートは、カスタムロードバランサー設定の設計または管理を支援することはできません。ソリューションは、組織のインフラストラクチャまたは運用チームによって精査、サポート、および維持される必要があります。

PEM 10.1 から、PEM エージェントと PEM Web アプリケーションの両方が、バックエンドデータベースに接続するためのマルチホスト接続文字列をサポートしています。マルチホスト接続文字列の場合

- クライアント PEM エージェントまたは Web アプリケーションは、書き込み接続を受け入れるまで、リストされた各ホストを試行します。
- フェールオーバー時に、接続がドロップされ、クライアントは再び検索を開始します。
- 文字列の最初のホストがプライマリでない場合、接続試行は失敗します。この障害により、プライマリへの接続を確立するまでにかかる時間が増加し、監視対象サーバーと PEM バックエンドサーバーでのリソースの使用量が増加します。

5.2.8.5 リファレンスアーキテクチャ

以下のすべてのアーキテクチャは、PEM バックエンドデータベースを備えた 3 つのノードを示しています。ただし、フェールオーバーマネージャーが偶数のノード番号を使用できる場合、このアプローチは 2 つのノードのみで動作します。そうでない場合は、完全な 3 番目のノードの代わりに監視ノードを使用できます。3 つを超えるノードも正常に動作しますが、その構成は、ほとんどの展開で大きな利点が得られない場合があります。

C1: 単一のエンドポイントとコロケーション

PEM Web アプリケーションとバックエンドデータベースの両方が同じホストで実行される最小限のセットアップ。

長所

- Web アプリケーションユーザーとモニタリングエージェントの両方の同じエンドポイントへの透過的な接続。

- すべてのインスタンスがループバックインターフェイスに接続するため、Web アプリケーションの構成とアップグレードが簡素化されます。
- 小規模な展開に適しています。

短所

- 共有リソースがボトルネックになる可能性があります。
- Web アプリケーションの障害に対する回復力がありません。フェールオーバー条件をカスタマイズしない限り、データベースに障害が発生するとフェールオーバーが発生します。
- Web アプリケーションは書き込み不可の Postgres サーバーへの接続を拒否するため、レプリカで実行されている Web アプリケーションは使用できません。

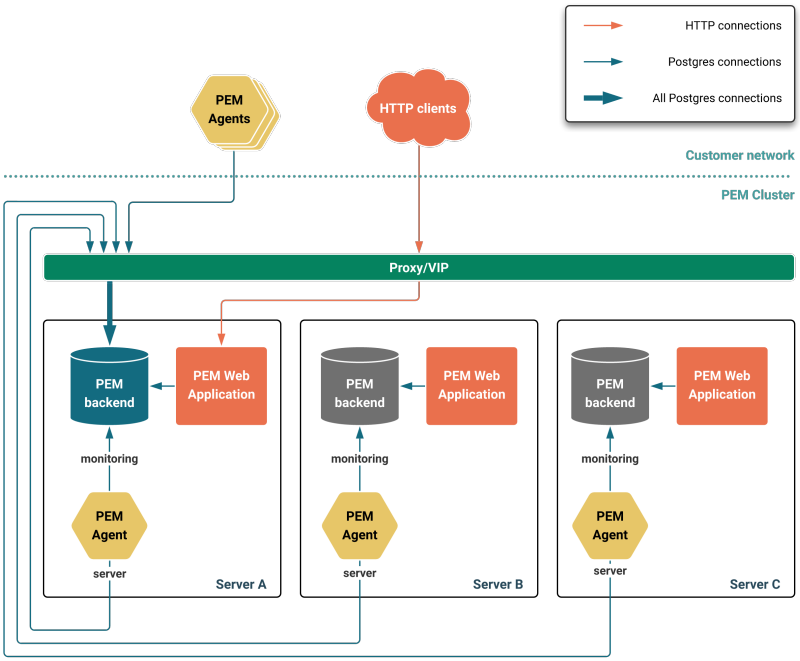


図 6: reference arch

S1 データベースの単一のエンドポイントで分離

PEM Web アプリケーションとバックエンドデータベースの両方が別のホストで実行されるセットアップ。

長所

- モニタリングエージェントは常に同じエンドポイントに接続するため、透過的な接続。
- Web アプリケーションとデータベースコンポーネントは、独立してスケーラブルで冗長です。
- 大規模な展開に適しています。

短所

- フロントエンドロードバランサーを使用しない限り、ユーザーは Web アプリケーションインスタンスを手動で選択する必要があります。

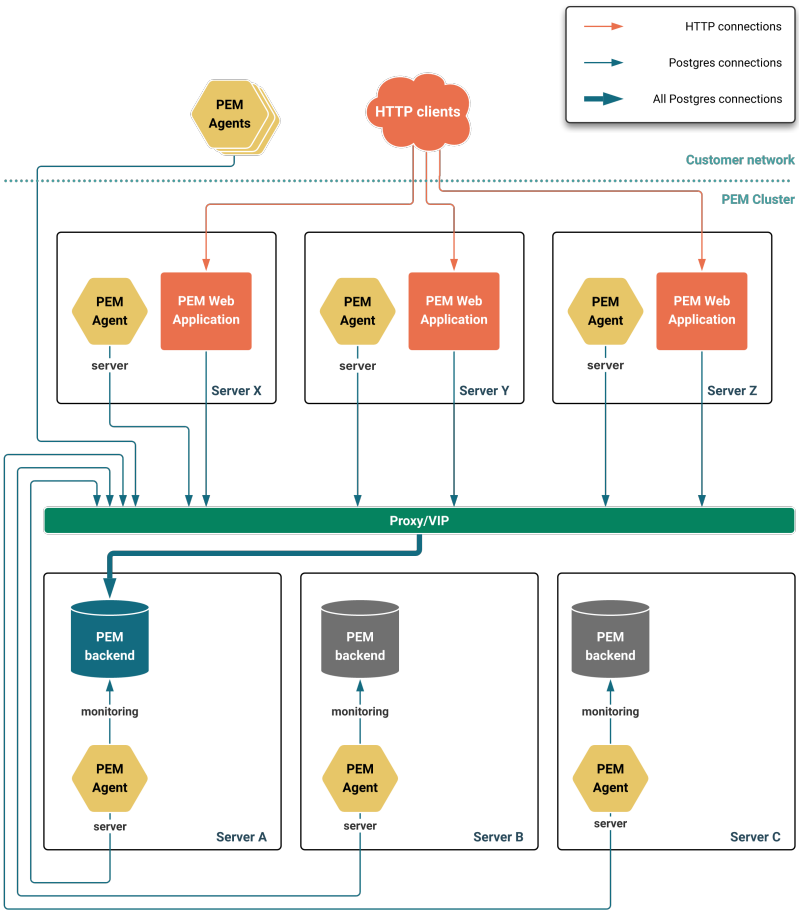


図 7: reference arch

CM マルチホスト接続文字列とのコロケーション

シングルエンドポイント設定は、マルチホスト接続文字列を使用するクライアントに置き換えられ、トラフィックがプライマリにルーティングされるようにします。Web アプリケーションとバックエンドデータベースは同じ場所にあります。

長所

- データベースサーバーのロードバランサーまたは VIP を管理する必要はありません。
- 冗長 Web アプリケーション。
- S1 および SM よりも小さいフットプリント。

短所

- すべてのモニタリングエージェントと Web アプリケーションを、すべてのバックエンドデータベースの接続の詳細で構成する必要があるため、再構成が困難。
- Web アプリケーションの複数のインスタンスを実行すると、混乱を招く場合があります。フロントエンドロードバランサーを使用しない限り、ユーザーは Web アプリケーションインスタンスを手動で選択する必要があります。
- マルチホスト接続は、単一のエンドポイントを介したルーティングよりも効率が低くなります。

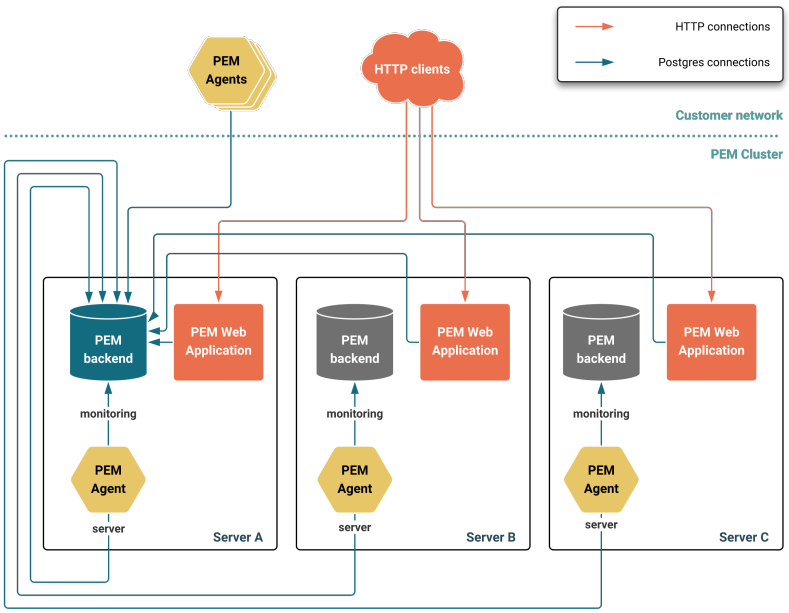


図 8: reference arch

SM マルチホスト接続文字列で区切られる

この設定では、単一のエンドポイントを削除しますが、Web アプリケーションとバックエンドデータベースを分離して、スケーラビリティを向上させます。

長所

- データベースサーバーのロードバランサーまたは VIP を管理する必要はありません。

- 冗長 Web アプリケーションとバックエンドデータベースコンポーネント。
- スケーラビリティを向上させるための大規模な展開に適しています。

短所

- すべてのモニタリングエージェントと Web アプリケーションを、すべてのバックエンドデータベースの接続の詳細で構成する必要があるため、再構成が困難。
- Web アプリケーションの複数のインスタンスを実行すると、混乱を招く場合があります。フロントエンドロードバランサーを使用している場合を除き、ユーザーは Web アプリケーションインスタンスを手動で選択する必要があります。
- マルチホスト接続は、単一のエンドポイントを介したルーティングよりも効率が低くなります

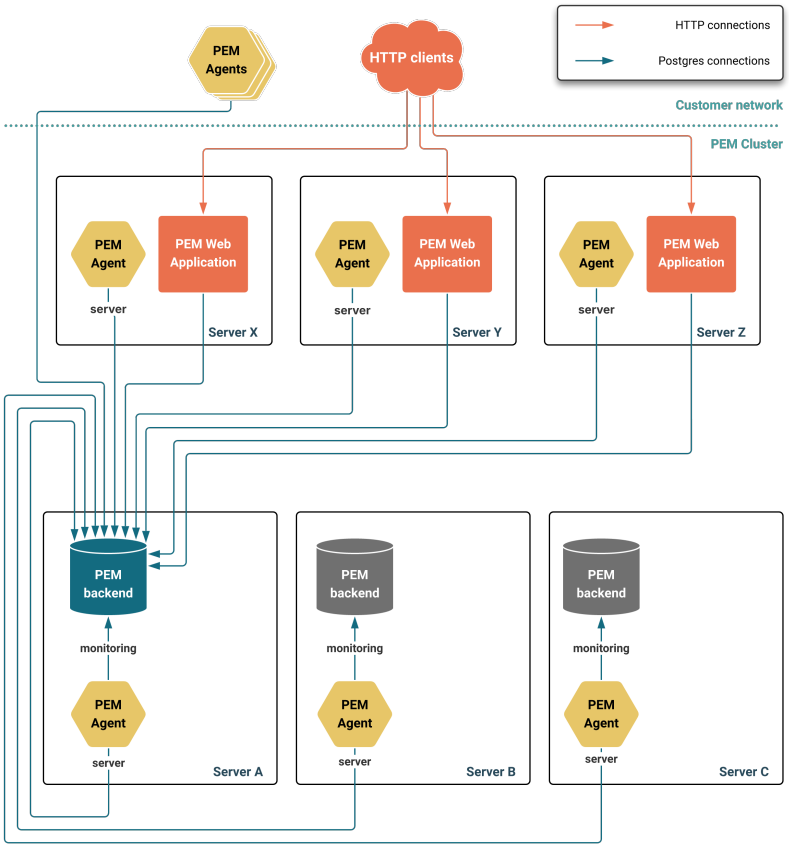


図 9: reference arch

第6章　INSTALLING AND UPGRADING

6.1 Installing Postgres Enterprise Manager server

リンクを選択して、該当するインストール手順にアクセスします。

6.1.1 Linux x86-64 (amd64)

6.1.1.1 Red Hat Enterprise Linux RHEL とその派生

- *Installing Postgres Enterprise Manager server on RHEL 10 or OL 10 x86_64*、*Installing Postgres Enterprise Manager server on RHEL 9 or OL 9 arm64*、 *Installing Postgres Enterprise Manager server on RHEL 8 ppc64le*
- *Installing Postgres Enterprise Manager server on RHEL 10 or OL 10 x86_64*、*Installing Postgres Enterprise Manager server on RHEL 9 or OL 9 arm64*、 *Installing Postgres Enterprise Manager server on RHEL 8 ppc64le*
- *Installing Postgres Enterprise Manager server on AlmaLinux 9 or Rocky*、 *Installing Postgres Enterprise Manager server on AlmaLinux 8 or Rocky*
- *Installing Postgres Enterprise Manager server on AlmaLinux 9 or Rocky*、 *Installing Postgres Enterprise Manager server on AlmaLinux 8 or Rocky*

6.1.1.2 SUSE Linux Enterprise SLES

- *Installing Postgres Enterprise Manager server on SLES 15 ppc64le*

6.1.1.3 Debian とデリバティブ

- *Installing Postgres Enterprise Manager server on Ubuntu 24.04 x86_64*、 *Installing Postgres Enterprise Manager server on Ubuntu 22.04 x86_64*
- *Installing Postgres Enterprise Manager server on Debian 12 arm64*、 *Installing Postgres Enterprise Manager server on Debian 11 x86_64*

6.1.2 Linux Installing Postgres Enterprise Manager server on Linux IBM Power (ppc64le)

6.1.2.1 Red Hat Enterprise Linux RHEL とその派生

- *Installing Postgres Enterprise Manager server on RHEL 9 or OL 9 arm64*、 *Installing Postgres Enterprise Manager server on RHEL 8 ppc64le*

6.1.2.2 SUSE Linux Enterprise SLES

- *Installing Postgres Enterprise Manager server on SLES 15 ppc64le*

6.1.3 Linux Installing Postgres Enterprise Manager server on Linux AArch64 (ARM64)

6.1.3.1 Red Hat Enterprise Linux RHEL とその派生

- *Installing Postgres Enterprise Manager server on RHEL 9 or OL 9 arm64*
- *Installing Postgres Enterprise Manager server on RHEL 9 or OL 9 arm64*

6.1.3.2 Debian とデリバティブ

- *Installing Postgres Enterprise Manager server on Debian 12 arm64*

6.1.4 Windows

- *Windows Server 2019*

6.1.4.1 Dependencies of the PEM Server and Agent on Linux

Linux の PEM サーバーおよびエージェントパッケージは、さまざまなシステムライブラリに依存しています。参考として、これらの依存関係について以下に詳しく説明します。

> **注釈**
>
> PEM エージェントは常に PEM サーバーとともにインストールされるため、すべての依存関係は PEM サーバーデータベースまたは Web アプリケーションがインストールされるホストに存在する必要があります。

通常、PEM は、特定のプラットフォームとアーキテクチャのベンダーリポジトリから入手可能な各依存関係の最新バージョンに対してビルドされます。場合によっては、PEM には、ベンダーリポジトリで利用可能なものよりも新しいバージョンのライブラリが必要です。これらの場合、パッケージの新しいバージョンの接頭辞 edb- は EDB のリポジトリから取得されます。

> **注釈**
>
> この情報は参考のために提供されています。ベンダーリポジトリからのパッケージは、EDB によってサポートまたはパッチされていません。 これらのパッケージの詳細については、オペレーティングシステムのドキュメントまたはサポートプロバイダーを参照してください。

これらの依存関係は頻繁に更新されるため、以下の表は PEM の最新パッチリリースでのみ有効です。

PEM サーバーには Python 3 が必要です。 Apache HTTPD を Web サーバーとして使用する場合、mod_wsgi モジュールも必要です。

Platform	Archi-tecture	Python/mod_wsgi package	Python version	Python path
RHEL 8	ppc64le	*edb-python312/edb-python312-mod-wsgi*	3.12	*/usr/libexec/edb-python312/bin/python3*
	s390x	*edb-python312/edb-python312-mod-wsgi*	3.12	*/usr/libexec/edb-python312/bin/python3*
	x86_64	*edb-python312/edb-python312-mod-wsgi*	3.12	*/usr/libexec/edb-python312/bin/python3*
RHEL 9	x86_64	*edb-python312/edb-python312-mod-wsgi*	3.12	*/usr/libexec/edb-python312/bin/python3*
	ppc64le	*edb-python312/edb-python312-mod-wsgi*	3.12	*/usr/libexec/edb-python312/bin/python3*
	s390x	*edb-python312/edb-python312-mod-wsgi*	3.12	*/usr/libexec/edb-python312/bin/python3*
	arm64	*edb-python312/edb-python312-mod-wsgi*	3.12	*/usr/libexec/edb-python312/bin/python3*
SLES 15	x86_64	*edb-python312/edb-python312-mod-wsgi*	3.12	*/usr/libexec/edb-python312/bin/python3*
	ppc64le	*edb-python312/edb-python312-mod-wsgi*	3.12	*/usr/libexec/edb-python312/bin/python3*
	s390x	*edb-python312/edb-python312-mod-wsgi*	3.12	*/usr/libexec/edb-python312/bin/python3*
Ubuntu 22	amd64	*edb-python312/edb-python312-mod-wsgi*	3.12	*/usr/libexec/edb-python312/bin/python3*
Ubuntu 24	amd64	*python312/libapache2-mod-wsgi-py3*	3.12	*/usr/bin/python3*
Debian 11	amd64	*edb-python312/edb-python312-mod-wsgi*	3.12	*/usr/libexec/edb-python312/bin/python3*
Debian 12	amd64/arm	*edb-python312/edb-python312-mod-wsgi*	3.12	*/usr/libexec/edb-python312/bin/python3*

OpenSSL

PEM サーバーとエージェントには OpenSSL が必要です。

Platform	Architecture	package-version
RHEL 8	x86_64	*openssl-1.1.1k*
	ppc64le	*openssl-1.1.1k*
	s390x	*openssl-1.1.1k*
RHEL 9	x86_64	*openssl-3.2.2*
	ppc64le	*openssl-3.2.2*
	s390x	*openssl-3.2.2*
	arm64	*openssl-3.2.2*

次のページに続く

表 2 – 前のページからの続き

Platform	Architecture	package-version
RHEL 10	x86_64	*openssl-3.5.1*
SLES 15	x86_64	*openssl-3.2.3*
	ppc64le	*openssl-3.2.3*
	s390x	*openssl-3.2.3*
Ubuntu 22	amd64	*openssl-3.0.2*
Ubuntu 24	amd64	*openssl-3.0.13*
Debian 11	amd64	*openssl-1.1.1w*
Debian 12	amd64/arm64	*openssl-3.0.17*

Libcurl

PEM エージェントには libcurl が必要です。

Platform	Architecture	package-version
RHEL 8	x86_64	*libcurl-7.61.1*
	ppc64le	*libcurl-7.61.1*
	s390x	*libcurl-7.61.1*
RHEL 9	x86_64	*libcurl-7.76.1/libcurl-minimal-7.76.1*
	ppc64le	*libcurl-7.76.1/libcurl-minimal-7.76.1*
	s390x	*libcurl-7.76.1/libcurl-minimal-7.76.1*
	arm64	*libcurl-7.76.1/libcurl-minimal-7.76.1*
RHEL 10	x86_64	*libcurl-8.12.1*
SLES 15	x86_64	*libcurl4-8.14.1*
	ppc64le	*libcurl4-8.14.1*
	s390x	*libcurl4-8.14.1*
Ubuntu 22	amd64	*libcurl4-7.81.0*
Ubuntu 24	amd64	*libcurl4t64-8.5.0*
Debian 11	amd64	*libcurl4-7.74.0*
Debian 12	amd64/arm64	*libcurl4-7.88.1*

SNMP++

PEM エージェントには SNMP++が必要です。

Platform	Architecture	package-version
RHEL 8	x86_64	*edb-snmp++-3.6.3*
	ppc64le	*edb-snmp++-3.6.3*

次のページに続く

表 4 – 前のページからの続き

Platform	Architecture	package-version
	s390x	*edb-snmp++-3.6.3*
RHEL 9	x86_64	*edb-snmp++-3.6.6*
	ppc64le	*edb-snmp++-3.6.6*
	s390x	*edb-snmp++-3.6.6*
	arm64	*edb-snmp++-3.6.6*
RHEL 10	x86_64	*edb-snmp++-3.6.6*
SLES 15	x86_64	*edb-snmp++-3.6.6*
	ppc64le	*edb-snmp++-3.6.6*
	s390x	*edb-snmp++-3.6.6*
Ubuntu 22	amd64	*edb-snmp++-3.6.6*
Ubuntu 24	amd64	*edb-snmp++-3.6.6*
Debian 11	amd64	*edb-snmp++-3.6.3*
Debian 12	amd64/arm64	*edb-snmp++-3.6.6*

Boost ライブラリ

PEM エージェントには、Boost ライブラリが必要です。

Platform	Architecture	package-version
RHEL 8	x86_64	*boost169-system-1.69.0*
	ppc64le	*boost-system-1.66.0*
	s390x	*boost-system-1.66.0*
RHEL 9	x86_64	*boost-system-1.75.0*
	ppc64le	*boost-system-1.75.0*
	s390x	*boost-system-1.75.0*
	arm64	*boost-system-1.75.0*
RHEL 10	x86_64	*boost-system-1.83.0*
SLES 15	x86_64	*libboost_filesystem1_66_1-1.66.0*
	ppc64le	*libboost_filesystem1_66_1-1.66.0*
	s390x	*libboost_filesystem1_66_1-1.66.0*
Ubuntu 22	amd64	*libboost-system1.74.0-1.74.0*
Ubuntu 24	amd64	*libboost-system1.74.0-1.74.0*
Debian 11	amd64	*libboost-system1.74.0-1.74.0*
Debian 12	amd64/arm64	*libboost-system1.74.0-1.74.0*

6.1.4.2 Installing Postgres Enterprise Manager server on Linux x86 (amd64)

オペレーティングシステム固有のインストール手順は、対応するドキュメントに記載されています。

Red Hat Enterprise Linux RHEL とその派生

- *Installing Postgres Enterprise Manager server on RHEL 10 or OL 10 x86_64*
- *Installing Postgres Enterprise Manager server on RHEL 9 or OL 9 arm64*
- *Installing Postgres Enterprise Manager server on RHEL 8 ppc64le*
- *Installing Postgres Enterprise Manager server on RHEL 10 or OL 10 x86_64*
- *Installing Postgres Enterprise Manager server on RHEL 9 or OL 9 arm64*
- *Installing Postgres Enterprise Manager server on RHEL 8 ppc64le*
- *Installing Postgres Enterprise Manager server on AlmaLinux 9 or Rocky*
- *Installing Postgres Enterprise Manager server on AlmaLinux 8 or Rocky*
- *Installing Postgres Enterprise Manager server on AlmaLinux 9 or Rocky*
- *Installing Postgres Enterprise Manager server on AlmaLinux 8 or Rocky*

SUSE Linux Enterprise SLES

- *Installing Postgres Enterprise Manager server on SLES 15 ppc64le*

Debian とデリバティブ

- *Installing Postgres Enterprise Manager server on Ubuntu 24.04 x86_64*
- *Installing Postgres Enterprise Manager server on Ubuntu 22.04 x86_64*
- *Installing Postgres Enterprise Manager server on Debian 12 arm64*
- *Installing Postgres Enterprise Manager server on Debian 11 x86_64*

Installing Postgres Enterprise Manager server on RHEL 10 or OL 10 x86_64

1 つのサーバーに PEM をインストールするか、Web アプリケーションサーバーとバックエンドデータベースを 2 つの別のサーバーにインストールできます。PEM インストールのためにサーバーを準備する必要があります。

前提条件を満たし、次の手順で説明するインストール手順を完了した後、*Web* **サーバー** *NGINX* **の構成** PEM を行う必要があります。2 つのサーバーを使用している場合は、両方のサーバーに PEM をインストールして構成します。

前提条件

インストールプロセスを開始する前に

1. バックエンドデータベースとして使用する PEM の supported Postgres instance をインストールします。

このインスタンスは、PEM Web アプリケーションに使用するのと同じサーバーにインストールすることも、別のサーバーにインストールすることもできます。次の手順で詳細に説明するように構成されている場合、既存の Postgres インスタンスを使用することもできます。

2. pg_hba.conf ファイルを更新することにより、Postgres バックエンドデータベースで認証を構成します。

構成する前に、次の変更を手動で加えます。*SELinux* **構成を変更する** の実行中に追加の変更が必要です。

- PEM に必要な関係を作成するために、PEM 構成スクリプトは、パスワード認証を使用して選択したスーパーユーザーとして Postgres バックエンドデータベースに接続します。これには、選択したスーパーユーザーにパスワードを使用した認証を許可する必要があります。このユーザーは、構成スクリプトを実行する任意の場所から接続できる必要があります。実際、これは、バックエンドデータベースが存在するサーバーと PEM Web アプリケーションがインストールされるサーバーが異なる場合を意味します。
- 選択したスーパーユーザーがパスワード認証を使用して接続できるようにするには、次の行を pg_hba.conf に追加します `host all superusername 127.0.0.1/32 scram-sha-256` などの `md5` または `scram-sha-256` 認証を使用した `host` 接続を許可します。

> **注釈**
>
> EDB Postgres Advanced Server を使用している場合、Modifying the pg_hba.conf file を参照してください。

PostgreSQL を使用している場合、 Client Authentication を参照してください。 .. ::
2_blank_1_3. sslutils 拡張機能が Postgres サーバーにインストールされていることを確認します。

RHEL/AlmaLinux/Rocky Linux または SLES で PostgreSQL または EDB Postgres Extended Server を使用している場合は、 `hstore contrib` モジュールをインストールする必要があります。

- EDB Postgres Advanced Server を使用している場合、次のように `sslutils` 拡張機能をインストールできます。 <x> は、 EDB Postgres Advanced サーバーバージョンです。

```
sudo dnf install edb-as<x>-server-sslutils
```

- PostgreSQL を使用している場合、次のように `sslutils` 、および必要に応じて `hstore` モジュールをインストールできます。<x> は PostgreSQL バージョンです。

```
sudo dnf install sslutils_<x> postgresql<x>-contrib
```

- EDB Postgres Extended Server を使用している場合、次のように `sslutils` 、および必要に応じて `hstore` モジュールをインストールできます。<x> は EDB Postgres Extended Server バージョンです。

```
sudo dnf install edb-postgresextended<x>-sslutils edb-postgresextended<x>-contrib
```

4. ファイアウォールを使用している場合は、PEM Web アプリケーションが配置されるサーバーでポート 8443 へのアクセスを許可します。

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

firewall-cmd --reload
```

5. Postgres Enterprise Manager が依存するコンポーネントがすべてのサーバーで最新の状態であることを確認します。これは、以下に示すようにパッケージマネージャーを使用してシステム全体を更新することにより実行できます。個々のパッケージを更新する場合、依存関係の完全なリストが *Dependencies of the PEM Server and Agent on Linux* で提供されています。

```
sudo dnf upgrade
```

パッケージをインストールする

```
sudo dnf -y install edb-pem
```

初期構成

```
#  You can configure the PEM server using the following command:

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

詳細については、 *Configuring the PEM server on Linux* を参照してください。

注釈

- オペレーティングシステムユーザー pem は、PEM サーバーのインストール中に作成されます。PEM アプリケーションデータとセッションは、このユーザーのホームディレクトリに保存されます。

サポートされているロケール

現在、Postgres Enterprise Manager サーバーと Web インターフェイスは、English(US) en_US ロケールと言語区切り文字としてのピリオド. の使用をサポートしています。代替ロケールまたはピリオド以外の区切り文字を使用すると、エラーが発生する場合があります。

Installing Postgres Enterprise Manager server on RHEL 9 or OL 9 x86_64

1 つのサーバーに PEM をインストールするか、Web アプリケーションサーバーとバックエンドデータベースを 2 つの別のサーバーにインストールできます。PEM インストールのためにサーバーを準備する必要があります。

前提条件を満たし、次の手順で説明するインストール手順を完了した後、*Web* サーバー *NGINX* の構成 PEM を行う必要があります。2 つのサーバーを使用している場合は、両方のサーバーに PEM をインストールして構成します。

前提条件

インストールプロセスを開始する前に

1. バックエンドデータベースとして使用する PEM の supported Postgres instance をインストールします。

このインスタンスは、PEM Web アプリケーションに使用するのと同じサーバーにインストールすることも、別のサーバーにインストールすることもできます。次の手順で詳細に説明するように構成されている場合、既存の Postgres インスタンスを使用することもできます。

2. `pg_hba.conf` ファイルを更新することにより、Postgres バックエンドデータベースで認証を構成します。

構成する前に、次の変更を手動で加えます。*SELinux* **構成を変更する** の実行中に追加の変更が必要です。

- PEM に必要な関係を作成するために、PEM 構成スクリプトは、パスワード認証を使用して選択したスーパーユーザーとして Postgres バックエンドデータベースに接続します。これには、選択したスーパーユーザーにパスワードを使用した認証を許可する必要があります。このユーザーは、構成スクリプトを実行する任意の場所から接続できる必要があります。実際、これは、バックエンドデータベースが存在するサーバーと PEM Web アプリケーションがインストールされるサーバーが異なる場合を意味します。
- 選択したスーパーユーザーがパスワード認証を使用して接続できるようにするには、次の行を `pg_hba.conf` に追加します `host all superusername 127.0.0.1/32 scram-sha-256` などの `md5` または `scram-sha-256` 認証を使用した `host` 接続を許可します。

> **注釈**
>
> EDB Postgres Advanced Server を使用している場合、Modifying the pg_hba.conf file を参照してください。

PostgreSQL を使用している場合、 Client Authentication を参照してください。 .. ::
2_blank_1_3. sslutils 拡張機能が Postgres サーバーにインストールされていることを確認します。

RHEL/AlmaLinux/Rocky Linux または SLES で PostgreSQL または EDB Postgres Extended Server を使用している場合は、 `hstore contrib` モジュールをインストールする必要があります。

- EDB Postgres Advanced Server を使用している場合、次のように `sslutils` 拡張機能をインストールできます。 <x> は、 EDB Postgres Advanced サーバーバージョンです。

```
sudo dnf install edb-as<x>-server-sslutils
```

- PostgreSQL を使用している場合、次のように `sslutils` 、および必要に応じて `hstore` モジュールをインストールできます。<x> は PostgreSQL バージョンです。

```
sudo dnf install sslutils_<x> postgresql<x>-contrib
```

- EDB Postgres Extended Server を使用している場合、次のように `sslutils` 、および必要に応じて `hstore` モジュールをインストールできます。<x> は EDB Postgres Extended Server バージョンです。

```
sudo dnf install edb-postgresextended<x>-sslutils edb-postgresextended<x>-contrib
```

4. ファイアウォールを使用している場合は、PEM Web アプリケーションが配置されるサーバーでポート 8443 へのアクセスを許可します。

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

firewall-cmd --reload
```

5. Postgres Enterprise Manager が依存するコンポーネントがすべてのサーバーで最新の状態であることを確認します。これは、以下に示すようにパッケージマネージャーを使用してシステム全体を更新することにより実行できます。個々のパッケージを更新する場合、依存関係の完全なリストが *Dependencies of the PEM Server and Agent on Linux* で提供されています。

```
sudo dnf upgrade
```

パッケージをインストールする

```
sudo dnf -y install edb-pem
```

初期構成

```
#  You can configure the PEM server using the following command:

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

詳細については、 *Configuring the PEM server on Linux* を参照してください。

注釈

- オペレーティングシステムユーザー pem は、PEM サーバーのインストール中に作成されます。PEM アプリケーションデータとセッションは、このユーザーのホームディレクトリに保存されます。

サポートされているロケール

現在、Postgres Enterprise Manager サーバーと Web インターフェイスは、English(US) en_US ロケールと言語区切り文字としてのピリオド. の使用をサポートしています。代替ロケールまたはピリオド以外の区切り文字を使用すると、エラーが発生する場合があります。

Installing Postgres Enterprise Manager server on RHEL 8 or OL 8 x86_64

1 つのサーバーに PEM をインストールするか、Web アプリケーションサーバーとバックエンドデータベースを 2 つの別のサーバーにインストールできます。PEM インストールのためにサーバーを準備する必要があります。

前提条件を満たし、次の手順で説明するインストール手順を完了した後、*Web* サーバー *NGINX* の構成 PEM を行う必要があります。2 つのサーバーを使用している場合は、両方のサーバーに PEM をインストールして構成します。

前提条件

インストールプロセスを開始する前に

1. バックエンドデータベースとして使用する PEM の supported Postgres instance をインストールします。

このインスタンスは、PEM Web アプリケーションに使用するのと同じサーバーにインストールすることも、別のサーバーにインストールすることもできます。次の手順で詳細に説明するように構成されている場合、既存の Postgres インスタンスを使用することもできます。

2. `pg_hba.conf` ファイルを更新することにより、Postgres バックエンドデータベースで認証を構成します。

構成する前に、次の変更を手動で加えます。*SELinux* **構成を変更する** の実行中に追加の変更が必要です。

- PEM に必要な関係を作成するために、PEM 構成スクリプトは、パスワード認証を使用して選択したスーパーユーザーとして Postgres バックエンドデータベースに接続します。これには、選択したスーパーユーザーにパスワードを使用した認証を許可する必要があります。このユーザーは、構成スクリプトを実行する任意の場所から接続できる必要があります。実際、これは、バックエンドデータベースが存在するサーバーと PEM Web アプリケーションがインストールされるサーバーが異なる場合を意味します。
- 選択したスーパーユーザーがパスワード認証を使用して接続できるようにするには、次の行を `pg_hba.conf` に追加します `host all superusername 127.0.0.1/32 scram-sha-256` などの `md5` または `scram-sha-256` 認証を使用した `host` 接続を許可します。

> **注釈**
>
> EDB Postgres Advanced Server を使用している場合、Modifying the pg_hba.conf file を参照してください。

PostgreSQL を使用している場合、 Client Authentication を参照してください。 .. ::
2_blank_1_3. sslutils 拡張機能が Postgres サーバーにインストールされていることを確認します。

RHEL/AlmaLinux/Rocky Linux または SLES で PostgreSQL または EDB Postgres Extended Server を使用している場合は、 `hstore contrib` モジュールをインストールする必要があります。

- EDB Postgres Advanced Server を使用している場合、次のように `sslutils` 拡張機能をインストールできます。 <x> は、 EDB Postgres Advanced サーバーバージョンです。

```
sudo dnf install edb-as<x>-server-sslutils
```

- PostgreSQL を使用している場合、次のように `sslutils` 、および必要に応じて `hstore` モジュールをインストールできます。<x> は PostgreSQL バージョンです。

```
sudo dnf install sslutils_<x> postgresql<x>-contrib
```

- EDB Postgres Extended Server を使用している場合、次のように `sslutils` 、および必要に応じて `hstore` モジュールをインストールできます。<x> は EDB Postgres Extended Server バージョンです。

```
sudo dnf install edb-postgresextended<x>-sslutils edb-postgresextended<x>-contrib
```

4. ファイアウォールを使用している場合は、PEM Web アプリケーションが配置されるサーバーでポート 8443 へのアクセスを許可します。

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

firewall-cmd --reload
```

5. Postgres Enterprise Manager が依存するコンポーネントがすべてのサーバーで最新の状態であることを確認します。これは、以下に示すようにパッケージマネージャーを使用してシステム全体を更新することにより実行できます。個々のパッケージを更新する場合、依存関係の完全なリストが *Dependencies of the PEM Server and Agent on Linux* で提供されています。

```
sudo dnf upgrade
```

パッケージをインストールする

```
sudo dnf -y install edb-pem
```

初期構成

```
#  You can configure the PEM server using the following command:

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

詳細については、 *Configuring the PEM server on Linux* を参照してください。

注釈

- オペレーティングシステムユーザー pem は、PEM サーバーのインストール中に作成されます。PEM アプリケーションデータとセッションは、このユーザーのホームディレクトリに保存されます。

サポートされているロケール

現在、Postgres Enterprise Manager サーバーと Web インターフェイスは、English(US) en_US ロケールと言語区切り文字としてのピリオド. の使用をサポートしています。代替ロケールまたはピリオド以外の区切り文字を使用すると、エラーが発生する場合があります。

Installing Postgres Enterprise Manager server on AlmaLinux 9 or Rocky

1 つのサーバーに PEM をインストールするか、Web アプリケーションサーバーとバックエンドデータベースを 2 つの別のサーバーにインストールできます。PEM インストールのためにサーバーを準備する必要があります。

前提条件を満たし、次の手順で説明するインストール手順を完了した後、*Web* サーバー *NGINX* の構成 PEM を行う必要があります。2 つのサーバーを使用している場合は、両方のサーバーに PEM をインストールして構成します。

前提条件

インストールプロセスを開始する前に

1. バックエンドデータベースとして使用する PEM の supported Postgres instance をインストールします。

このインスタンスは、PEM Web アプリケーションに使用するのと同じサーバーにインストールすることも、別のサーバーにインストールすることもできます。次の手順で詳細に説明するように構成されている場合、既存の Postgres インスタンスを使用することもできます。

2. `pg_hba.conf` ファイルを更新することにより、Postgres バックエンドデータベースで認証を構成します。

構成する前に、次の変更を手動で加えます。*SELinux* **構成を変更する** の実行中に追加の変更が必要です。

- PEM に必要な関係を作成するために、PEM 構成スクリプトは、パスワード認証を使用して選択したスーパーユーザーとして Postgres バックエンドデータベースに接続します。これには、選択したスーパーユーザーにパスワードを使用した認証を許可する必要があります。このユーザーは、構成スクリプトを実行する任意の場所から接続できる必要があります。実際、これは、バックエンドデータベースが存在するサーバーと PEM Web アプリケーションがインストールされるサーバーが異なる場合を意味します。
- 選択したスーパーユーザーがパスワード認証を使用して接続できるようにするには、次の行を `pg_hba.conf` に追加します `host all superusername 127.0.0.1/32 scram-sha-256` などの `md5` または `scram-sha-256` 認証を使用した `host` 接続を許可します。

> **注釈**
>
> EDB Postgres Advanced Server を使用している場合、Modifying the pg_hba.conf file を参照してください。

PostgreSQL を使用している場合、 Client Authentication を参照してください。 .. ::
2_blank_1_3. sslutils 拡張機能が Postgres サーバーにインストールされていることを確認します。

RHEL/AlmaLinux/Rocky Linux または SLES で PostgreSQL または EDB Postgres Extended Server を使用している場合は、 `hstore contrib` モジュールをインストールする必要があります。

- EDB Postgres Advanced Server を使用している場合、次のように `sslutils` 拡張機能をインストールできます。 <x> は、 EDB Postgres Advanced サーバーバージョンです。

```
sudo dnf install edb-as<x>-server-sslutils
```

- PostgreSQL を使用している場合、次のように `sslutils` 、および必要に応じて `hstore` モジュールをインストールできます。<x> は PostgreSQL バージョンです。

```
sudo dnf install sslutils_<x> postgresql<x>-contrib
```

- EDB Postgres Extended Server を使用している場合、次のように `sslutils` 、および必要に応じて `hstore` モジュールをインストールできます。<x> は EDB Postgres Extended Server バージョンです。

```
sudo dnf install edb-postgresextended<x>-sslutils edb-postgresextended<x>-contrib
```

4. ファイアウォールを使用している場合は、PEM Web アプリケーションが配置されるサーバーでポート 8443 へのアクセスを許可します。

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

firewall-cmd --reload
```

5. Postgres Enterprise Manager が依存するコンポーネントがすべてのサーバーで最新の状態であることを確認します。これは、以下に示すようにパッケージマネージャーを使用してシステム全体を更新することにより実行できます。個々のパッケージを更新する場合、依存関係の完全なリストが *Dependencies of the PEM Server and Agent on Linux* で提供されています。

```
sudo dnf upgrade
```

パッケージをインストールする

```
sudo dnf -y install edb-pem
```

初期構成

```
#  You can configure the PEM server using the following command:

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

詳細については、 *Configuring the PEM server on Linux* を参照してください。

注釈

- オペレーティングシステムユーザー pem は、PEM サーバーのインストール中に作成されます。PEM アプリケーションデータとセッションは、このユーザーのホームディレクトリに保存されます。

サポートされているロケール

現在、Postgres Enterprise Manager サーバーと Web インターフェイスは、English(US) en_US ロケールと言語区切り文字としてのピリオド. の使用をサポートしています。代替ロケールまたはピリオド以外の区切り文字を使用すると、エラーが発生する場合があります。

Installing Postgres Enterprise Manager server on AlmaLinux 8 or Rocky

1 つのサーバーに PEM をインストールするか、Web アプリケーションサーバーとバックエンドデータベースを 2 つの別のサーバーにインストールできます。PEM インストールのためにサーバーを準備する必要があります。

前提条件を満たし、次の手順で説明するインストール手順を完了した後、*Web* サーバー *NGINX* の構成 PEM を行う必要があります。2 つのサーバーを使用している場合は、両方のサーバーに PEM をインストールして構成します。

前提条件

インストールプロセスを開始する前に

1. バックエンドデータベースとして使用する PEM の supported Postgres instance をインストールします。

このインスタンスは、PEM Web アプリケーションに使用するのと同じサーバーにインストールすることも、別のサーバーにインストールすることもできます。次の手順で詳細に説明するように構成されている場合、既存の Postgres インスタンスを使用することもできます。

2. `pg_hba.conf` ファイルを更新することにより、Postgres バックエンドデータベースで認証を構成します。

構成する前に、次の変更を手動で加えます。*SELinux* **構成を変更する** の実行中に追加の変更が必要です。

- PEM に必要な関係を作成するために、PEM 構成スクリプトは、パスワード認証を使用して選択したスーパーユーザーとして Postgres バックエンドデータベースに接続します。これには、選択したスーパーユーザーにパスワードを使用した認証を許可する必要があります。このユーザーは、構成スクリプトを実行する任意の場所から接続できる必要があります。実際、これは、バックエンドデータベースが存在するサーバーと PEM Web アプリケーションがインストールされるサーバーが異なる場合を意味します。
- 選択したスーパーユーザーがパスワード認証を使用して接続できるようにするには、次の行を `pg_hba.conf` に追加します `host all superusername 127.0.0.1/32 scram-sha-256` などの `md5` または `scram-sha-256` 認証を使用した `host` 接続を許可します。

> **注釈**
>
> EDB Postgres Advanced Server を使用している場合、Modifying the pg_hba.conf file を参照してください。

PostgreSQL を使用している場合、 Client Authentication を参照してください。 .. ::
2_blank_1_3. sslutils 拡張機能が Postgres サーバーにインストールされていることを確認します。

RHEL/AlmaLinux/Rocky Linux または SLES で PostgreSQL または EDB Postgres Extended Server を使用している場合は、 `hstore contrib` モジュールをインストールする必要があります。

- EDB Postgres Advanced Server を使用している場合、次のように `sslutils` 拡張機能をインストールできます。 <x> は、 EDB Postgres Advanced サーバーバージョンです。

```
sudo dnf install edb-as<x>-server-sslutils
```

- PostgreSQL を使用している場合、次のように `sslutils` 、および必要に応じて `hstore` モジュールをインストールできます。<x> は PostgreSQL バージョンです。

```
sudo dnf install sslutils_<x> postgresql<x>-contrib
```

- EDB Postgres Extended Server を使用している場合、次のように `sslutils` 、および必要に応じて `hstore` モジュールをインストールできます。<x> は EDB Postgres Extended Server バージョンです。

```
sudo dnf install edb-postgresextended<x>-sslutils edb-postgresextended<x>-contrib
```

4. ファイアウォールを使用している場合は、PEM Web アプリケーションが配置されるサーバーでポート 8443 へのアクセスを許可します。

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

firewall-cmd --reload
```

5. Postgres Enterprise Manager が依存するコンポーネントがすべてのサーバーで最新の状態であることを確認します。これは、以下に示すようにパッケージマネージャーを使用してシステム全体を更新することにより実行できます。個々のパッケージを更新する場合、依存関係の完全なリストが *Dependencies of the PEM Server and Agent on Linux* で提供されています。

```
sudo dnf upgrade
```

パッケージをインストールする

```
sudo dnf -y install edb-pem
```

初期構成

```
#  You can configure the PEM server using the following command:

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

詳細については、 *Configuring the PEM server on Linux* を参照してください。

注釈

- オペレーティングシステムユーザー pem は、PEM サーバーのインストール中に作成されます。PEM アプリケーションデータとセッションは、このユーザーのホームディレクトリに保存されます。

サポートされているロケール

現在、Postgres Enterprise Manager サーバーと Web インターフェイスは、English(US) en_US ロケールと言語区切り文字としてのピリオド. の使用をサポートしています。代替ロケールまたはピリオド以外の区切り文字を使用すると、エラーが発生する場合があります。

Installing Postgres Enterprise Manager server on SLES 15 x86_64

注釈

Postgres Enterprise Manager 8.3 以降は SLES でサポートされています。

前提条件

インストールプロセスを開始する前に

1. バックエンドデータベースとして使用する PEM の supported Postgres instance をインストールします。

このインスタンスは、PEM Web アプリケーションに使用するのと同じサーバーにインストールすることも、別のサーバーにインストールすることもできます。次の手順で詳細に説明するように構成されている場合、既存の Postgres インスタンスを使用することもできます。

2. `pg_hba.conf` ファイルを更新することにより、Postgres バックエンドデータベースで認証を構成します。

構成する前に、次の変更を手動で加えます。*SELinux* **構成を変更する** の実行中に追加の変更が必要です。

- PEM に必要な関係を作成するために、PEM 構成スクリプトは、パスワード認証を使用して選択したスーパーユーザーとして Postgres バックエンドデータベースに接続します。これには、選択したスーパーユーザーにパスワードを使用した認証を許可する必要があります。このユーザーは、構成スクリプトを実行する任意の場所から接続できる必要があります。実際、これは、バックエンドデータベースが存在するサーバーと PEM Web アプリケーションがインストールされるサーバーが異なる場合を意味します。
- 選択したスーパーユーザーがパスワード認証を使用して接続できるようにするには、次の行を `pg_hba.conf` に追加します `host all superusername 127.0.0.1/32 scram-sha-256` などの `md5` または `scram-sha-256` 認証を使用した `host` 接続を許可します。

> **注釈**
>
> EDB Postgres Advanced Server を使用している場合、Modifying the pg_hba.conf file を参照してください。

PostgreSQL を使用している場合、 Client Authentication を参照してください。 .. ::
2_blank_1_3. sslutils 拡張機能が Postgres サーバーにインストールされていることを確認します。

RHEL/AlmaLinux/Rocky Linux または SLES で PostgreSQL または EDB Postgres Extended Server を使用している場合は、 `hstore contrib` モジュールをインストールする必要があります。

- EDB Postgres Advanced Server を使用している場合、次のように `sslutils` 拡張機能をインストールできます。 <x> は、 EDB Postgres Advanced サーバーバージョンです。

```
sudo zypper install edb-as<x>-server-sslutils
```

- PostgreSQL を使用している場合、次のように `sslutils` 、および必要に応じて `hstore` モジュールをインストールできます。<x> は PostgreSQL バージョンです。

```
sudo zypper install sslutils_<x> postgresql<x>-contrib
```

- EDB Postgres Extended Server を使用している場合、次のように `sslutils` 、および必要に応じて `hstore` モジュールをインストールできます。<x> は EDB Postgres Extended Server バージョンです。

```
sudo zypper install edb-postgresextended<x>-sslutils edb-postgresextended<x>-contrib
```

4. ファイアウォールを使用している場合は、PEM Web アプリケーションが配置されるサーバーでポート 8443 へのアクセスを許可します。

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

firewall-cmd --reload
```

5. Postgres Enterprise Manager が依存するコンポーネントがすべてのサーバーで最新の状態であることを確認します。これは、以下に示すようにパッケージマネージャーを使用してシステム全体を更新することにより実行できます。個々のパッケージを更新する場合、依存関係の完全なリストが *Dependencies of the PEM Server and Agent on Linux* で提供されています。

```
sudo zypper update
```

パッケージをインストールする

```
sudo zypper -n install edb-pem
```

初期構成

```
#  You can configure the PEM server using the following command:

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

詳細については、 *Configuring the PEM server on Linux* を参照してください。

注釈

- オペレーティングシステムユーザー pem は、PEM サーバーのインストール中に作成されます。PEM アプリケーションデータとセッションは、このユーザーのホームディレクトリに保存されます。

サポートされているロケール

現在、Postgres Enterprise Manager サーバーと Web インターフェイスは、English(US) en_US ロケールと言語区切り文字としてのピリオド. の使用をサポートしています。代替ロケールまたはピリオド以外の区切り文字を使用すると、エラーが発生する場合があります。

Installing Postgres Enterprise Manager server on Ubuntu 24.04 x86_64

1 つのサーバーに PEM をインストールするか、Web アプリケーションサーバーとバックエンドデータベースを 2 つの別のサーバーにインストールできます。PEM インストールのためにサーバーを準備する必要があります。

前提条件を満たし、次の手順で説明するインストール手順を完了した後、*Web* サーバー *NGINX* の構成 PEM を行う必要があります。2 つのサーバーを使用している場合は、両方のサーバーに PEM をインストールして構成します。

前提条件

インストールプロセスを開始する前に

1. バックエンドデータベースとして使用する PEM の supported Postgres instance をインストールします。

このインスタンスは、PEM Web アプリケーションに使用するのと同じサーバーにインストールすることも、別のサーバーにインストールすることもできます。次の手順で詳細に説明するように構成されている場合、既存の Postgres インスタンスを使用することもできます。

2. `pg_hba.conf` ファイルを更新することにより、Postgres バックエンドデータベースで認証を構成します。

構成する前に、次の変更を手動で加えます。*SELinux* **構成を変更する** の実行中に追加の変更が必要です。

- PEM に必要な関係を作成するために、PEM 構成スクリプトは、パスワード認証を使用して選択したスーパーユーザーとして Postgres バックエンドデータベースに接続します。これには、選択したスーパーユーザーにパスワードを使用した認証を許可する必要があります。このユーザーは、構成スクリプトを実行する任意の場所から接続できる必要があります。実際、これは、バックエンドデータベースが存在するサーバーと PEM Web アプリケーションがインストールされるサーバーが異なる場合を意味します。
- 選択したスーパーユーザーがパスワード認証を使用して接続できるようにするには、次の行を `pg_hba.conf` に追加します `host all superusername 127.0.0.1/32 scram-sha-256` などの `md5` または `scram-sha-256` 認証を使用した `host` 接続を許可します。

> **注釈**
>
> EDB Postgres Advanced Server を使用している場合、Modifying the pg_hba.conf file を参照してください。

PostgreSQL を使用している場合、 Client Authentication を参照してください。 .. ::
2_blank_1_3. sslutils 拡張機能が Postgres サーバーにインストールされていることを確認します。

RHEL/AlmaLinux/Rocky Linux または SLES で PostgreSQL または EDB Postgres Extended Server を使用している場合は、 `hstore contrib` モジュールをインストールする必要があります。

- EDB Postgres Advanced Server を使用している場合、次のように `sslutils` 拡張機能をインストールできます。 <x> は、 EDB Postgres Advanced サーバーバージョンです。

```
sudo apt-get install edb-as<x>-server-sslutils
```

- PostgreSQL を使用している場合、次のように `sslutils` 、および必要に応じて `hstore` モジュールをインストールできます。<x> は PostgreSQL バージョンです。

```
sudo apt-get install postgresql-<x>-sslutils
```

- EDB Postgres Extended Server を使用している場合、次のように `sslutils` 、および必要に応じて `hstore` モジュールをインストールできます。<x> は EDB Postgres Extended Server バージョンです。

```
sudo apt-get install edb-postgresextended-sslutils-<x>
```

> **注釈**
>
> Ubuntu 20 では、証明書を受け入れるための要件が変更されました。

- sslutils の古いバージョンを使用するマシンに PEM エージェントをインストールする場合は、sslutils を 1.3 にアップグレードする必要があります。バージョン 1.3 には、4096 ビット RSA キーと sha256 署名アルゴリズムのサポートが追加されています。- sslutils を 1.3 にアップグレードしない場合、PEM エージェントは PEM バックエンドデータベースサーバーへの接続に失敗し、エラー「ca md too promote.」ログに記録される場合があります。

```
iptables -t filter -A INPUT -p TCP --dport 8443 -j ACCEPT
```

5. Postgres Enterprise Manager が依存するコンポーネントがすべてのサーバーで最新の状態であることを確認します。これは、以下に示すようにパッケージマネージャーを使用してシステム全体を更新することにより実行できます。個々のパッケージを更新する場合、依存関係の完全なリストが *Dependencies of the PEM Server and Agent on Linux* で提供されています。

```
sudo apt-get update
```

パッケージをインストールする

```
sudo apt-get -y install edb-pem
```

初期構成

```
#  You can configure the PEM server using the following command:

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

詳細については、 *Configuring the PEM server on Linux* を参照してください。

> **注釈**
>
> - オペレーティングシステムユーザー pem は、PEM サーバーのインストール中に作成されます。PEM アプリケーションデータとセッションは、このユーザーのホームディレクトリに保存されます。

サポートされているロケール

現在、Postgres Enterprise Manager サーバーと Web インターフェイスは、English(US) en_US ロケールと言語区切り文字としてのピリオド. の使用をサポートしています。代替ロケールまたはピリオド以外の区切り文字を使用すると、エラーが発生する場合があります。

Installing Postgres Enterprise Manager server on Ubuntu 22.04 x86_64

1 つのサーバーに PEM をインストールするか、Web アプリケーションサーバーとバックエンドデータベースを 2 つの別のサーバーにインストールできます。PEM インストールのためにサーバーを準備する必要があります。

前提条件を満たし、次の手順で説明するインストール手順を完了した後、*Web* サーバー *NGINX* の構成 PEM を行う必要があります。2 つのサーバーを使用している場合は、両方のサーバーに PEM をインストールして構成します。

前提条件

インストールプロセスを開始する前に

1. バックエンドデータベースとして使用する PEM の supported Postgres instance をインストールします。

このインスタンスは、PEM Web アプリケーションに使用するのと同じサーバーにインストールすることも、別のサーバーにインストールすることもできます。次の手順で詳細に説明するように構成されている場合、既存の Postgres インスタンスを使用することもできます。

2. `pg_hba.conf` ファイルを更新することにより、Postgres バックエンドデータベースで認証を構成します。

構成する前に、次の変更を手動で加えます。*SELinux* 構成を変更する の実行中に追加の変更が必要です。

- PEM に必要な関係を作成するために、PEM 構成スクリプトは、パスワード認証を使用して選択したスーパーユーザーとして Postgres バックエンドデータベースに接続します。これには、選択したスーパーユーザーにパスワードを使用した認証を許可する必要があります。このユーザーは、構成スクリプトを実行する任意の場所から接続できる必要があります。実際、これは、バックエンドデータベースが存在するサーバーと PEM Web アプリケーションがインストールされるサーバーが異なる場合を意味します。
- 選択したスーパーユーザーがパスワード認証を使用して接続できるようにするには、次の行を `pg_hba.conf` に追加します `host all superusername 127.0.0.1/32 scram-sha-256` などの `md5` または `scram-sha-256` 認証を使用した `host` 接続を許可します。

> **注釈**
>
> EDB Postgres Advanced Server を使用している場合、Modifying the pg_hba.conf file を参照してください。

PostgreSQL を使用している場合、 Client Authentication を参照してください。 .. ::

2_blank_1_3. sslutils 拡張機能が Postgres サーバーにインストールされていることを確認します。

RHEL/AlmaLinux/Rocky Linux または SLES で PostgreSQL または EDB Postgres Extended Server を使用している場合は、 `hstore contrib` モジュールをインストールする必要があります。

- EDB Postgres Advanced Server を使用している場合、次のように `sslutils` 拡張機能をインストールできます。 <x> は、 EDB Postgres Advanced サーバーバージョンです。

```
sudo apt-get install edb-as<x>-server-sslutils
```

- PostgreSQL を使用している場合、次のように sslutils 、および必要に応じて hstore モジュールをインストールできます。<x> は PostgreSQL バージョンです。

```
sudo apt-get install postgresql-<x>-sslutils
```

- EDB Postgres Extended Server を使用している場合、次のように sslutils 、および必要に応じて hstore モジュールをインストールできます。<x> は EDB Postgres Extended Server バージョンです。

```
sudo apt-get install edb-postgresextended-sslutils-<x>
```

4. ファイアウォールを使用している場合は、PEM Web アプリケーションが配置されるサーバーでポート 8443 へのアクセスを許可します。

```
iptables -t filter -A INPUT -p TCP --dport 8443 -j ACCEPT
```

5. Postgres Enterprise Manager が依存するコンポーネントがすべてのサーバーで最新の状態であることを確認します。これは、以下に示すようにパッケージマネージャーを使用してシステム全体を更新することにより実行できます。個々のパッケージを更新する場合、依存関係の完全なリストが *Dependencies of the PEM Server and Agent on Linux* で提供されています。

```
sudo apt-get update
```

パッケージをインストールする

```
sudo apt-get -y install edb-pem
```

初期構成

```
#  You can configure the PEM server using the following command:

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

詳細については、 *Configuring the PEM server on Linux* を参照してください。

注釈

- オペレーティングシステムユーザー pem は、PEM サーバーのインストール中に作成されます。PEM アプリケーションデータとセッションは、このユーザーのホームディレクトリに保存されます。

サポートされているロケール

現在、Postgres Enterprise Manager サーバーと Web インターフェイスは、English(US) en_US ロケールと言語区切り文字としてのピリオド. の使用をサポートしています。代替ロケールまたはピリオド以外の区切り文字を使用すると、エラーが発生する場合があります。

Installing Postgres Enterprise Manager server on Debian 12 x86_64

1 つのサーバーに PEM をインストールするか、Web アプリケーションサーバーとバックエンドデータベースを 2 つの別のサーバーにインストールできます。PEM インストールのためにサーバーを準備する必要があります。

前提条件を満たし、次の手順で説明するインストール手順を完了した後、*Web サーバー NGINX の構成* PEM を行う必要があります。2 つのサーバーを使用している場合は、両方のサーバーに PEM をインストールして構成します。

前提条件

インストールプロセスを開始する前に

1. バックエンドデータベースとして使用する PEM の supported Postgres instance をインストールします。

このインスタンスは、PEM Web アプリケーションに使用するのと同じサーバーにインストールすることも、別のサーバーにインストールすることもできます。次の手順で詳細に説明するように構成されている場合、既存の Postgres インスタンスを使用することもできます。

2. `pg_hba.conf` ファイルを更新することにより、Postgres バックエンドデータベースで認証を構成します。

構成する前に、次の変更を手動で加えます。*SELinux 構成を変更する* の実行中に追加の変更が必要です。

- PEM に必要な関係を作成するために、PEM 構成スクリプトは、パスワード認証を使用して選択したスーパーユーザーとして Postgres バックエンドデータベースに接続します。これには、選択したスーパーユーザーにパスワードを使用した認証を許可する必要があります。このユーザーは、構成スクリプトを実行する任意の場所から接続できる必要があります。実際、これは、バックエンドデータベースが存在するサーバーと PEM Web アプリケーションがインストールされるサーバーが異なる場合を意味します。
- 選択したスーパーユーザーがパスワード認証を使用して接続できるようにするには、次の行を `pg_hba.conf` に追加します `host all superusername 127.0.0.1/32 scram-sha-256` などの `md5` または `scram-sha-256` 認証を使用した `host` 接続を許可します。

> **注釈**
>
> EDB Postgres Advanced Server を使用している場合、Modifying the pg_hba.conf file を参照してください。

PostgreSQL を使用している場合、 Client Authentication を参照してください。 .. ::

2_blank_1_3. sslutils 拡張機能が Postgres サーバーにインストールされていることを確認します。

RHEL/AlmaLinux/Rocky Linux または SLES で PostgreSQL または EDB Postgres Extended Server を使用している場合は、 `hstore contrib` モジュールをインストールする必要があります。

- EDB Postgres Advanced Server を使用している場合、次のように `sslutils` 拡張機能をインストールできます。 <x> は、 EDB Postgres Advanced サーバーバージョンです。

```
sudo apt-get install edb-as<x>-server-sslutils
```

- PostgreSQL を使用している場合、次のように sslutils 、および必要に応じて hstore モジュールをインストールできます。<x> は PostgreSQL バージョンです。

```
sudo apt-get install postgresql-<x>-sslutils
```

- EDB Postgres Extended Server を使用している場合、次のように sslutils 、および必要に応じて hstore モジュールをインストールできます。<x> は EDB Postgres Extended Server バージョンです。

```
sudo apt-get install edb-postgresextended-sslutils-<x>
```

4. ファイアウォールを使用している場合は、PEM Web アプリケーションが配置されるサーバーでポート 8443 へのアクセスを許可します。

5. Postgres Enterprise Manager が依存するコンポーネントがすべてのサーバーで最新の状態であることを確認します。これは、以下に示すようにパッケージマネージャーを使用してシステム全体を更新することにより実行できます。個々のパッケージを更新する場合、依存関係の完全なリストが *Dependencies of the PEM Server and Agent on Linux* で提供されています。

```
sudo apt-get upgrade
```

パッケージをインストールする

```
sudo apt-get -y install edb-pem
```

初期構成

```
#  You can configure the PEM server using the following command:

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

詳細については、 *Configuring the PEM server on Linux* を参照してください。

注釈

- オペレーティングシステムユーザー pem は、PEM サーバーのインストール中に作成されます。PEM アプリケーションデータとセッションは、このユーザーのホームディレクトリに保存されます。

サポートされているロケール

現在、Postgres Enterprise Manager サーバーと Web インターフェイスは、English(US) en_US ロケールと言語区切り文字としてのピリオド. の使用をサポートしています。代替ロケールまたはピリオド以外の区切り文字を使用すると、エラーが発生する場合があります。

Installing Postgres Enterprise Manager server on Debian 11 x86_64

1 つのサーバーに PEM をインストールするか、Web アプリケーションサーバーとバックエンドデータベースを 2 つの別のサーバーにインストールできます。PEM インストールのためにサーバーを準備する必要があります。

前提条件を満たし、次の手順で説明するインストール手順を完了した後、*Web* サーバー *NGINX* の構成 PEM を行う必要があります。2 つのサーバーを使用している場合は、両方のサーバーに PEM をインストールして構成します。

前提条件

インストールプロセスを開始する前に

1. バックエンドデータベースとして使用する PEM の supported Postgres instance をインストールします。

このインスタンスは、PEM Web アプリケーションに使用するのと同じサーバーにインストールすることも、別のサーバーにインストールすることもできます。次の手順で詳細に説明するように構成されている場合、既存の Postgres インスタンスを使用することもできます。

2. `pg_hba.conf` ファイルを更新することにより、Postgres バックエンドデータベースで認証を構成します。

構成する前に、次の変更を手動で加えます。*SELinux* 構成を変更する の実行中に追加の変更が必要です。

- PEM に必要な関係を作成するために、PEM 構成スクリプトは、パスワード認証を使用して選択したスーパーユーザーとして Postgres バックエンドデータベースに接続します。これには、選択したスーパーユーザーにパスワードを使用した認証を許可する必要があります。このユーザーは、構成スクリプトを実行する任意の場所から接続できる必要があります。実際、これは、バックエンドデータベースが存在するサーバーと PEM Web アプリケーションがインストールされるサーバーが異なる場合を意味します。
- 選択したスーパーユーザーがパスワード認証を使用して接続できるようにするには、次の行を `pg_hba.conf` に追加します `host all superusername 127.0.0.1/32 scram-sha-256` などの `md5` または `scram-sha-256` 認証を使用した `host` 接続を許可します。

> **注釈**
>
> EDB Postgres Advanced Server を使用している場合、Modifying the pg_hba.conf file を参照してください。

PostgreSQL を使用している場合、 Client Authentication を参照してください。 .. ::
2_blank_1_3. sslutils 拡張機能が Postgres サーバーにインストールされていることを確認します。

RHEL/AlmaLinux/Rocky Linux または SLES で PostgreSQL または EDB Postgres Extended Server を使用している場合は、 `hstore contrib` モジュールをインストールする必要があります。

- EDB Postgres Advanced Server を使用している場合、次のように `sslutils` 拡張機能をインストールできます。 <x> は、 EDB Postgres Advanced サーバーバージョンです。

```
sudo apt-get install edb-as<x>-server-sslutils
```

- PostgreSQL を使用している場合、次のように sslutils 、および必要に応じて hstore モジュールをインストールできます。<x> は PostgreSQL バージョンです。

```
sudo apt-get install postgresql-<x>-sslutils
```

- EDB Postgres Extended Server を使用している場合、次のように sslutils 、および必要に応じて hstore モジュールをインストールできます。<x> は EDB Postgres Extended Server バージョンです。

```
sudo apt-get install edb-postgresextended-sslutils-<x>
```

4. ファイアウォールを使用している場合は、PEM Web アプリケーションが配置されるサーバーでポート 8443 へのアクセスを許可します。

```
iptables -t filter -A INPUT -p TCP --dport 8443 -j ACCEPT
```

5. Postgres Enterprise Manager が依存するコンポーネントがすべてのサーバーで最新の状態であることを確認します。これは、以下に示すようにパッケージマネージャーを使用してシステム全体を更新することにより実行できます。個々のパッケージを更新する場合、依存関係の完全なリストが *Dependencies of the PEM Server and Agent on Linux* で提供されています。

```
sudo apt-get update
```

パッケージをインストールする

```
sudo apt-get -y install edb-pem
```

初期構成

```
#  You can configure the PEM server using the following command:

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

詳細については、 *Configuring the PEM server on Linux* を参照してください。

注釈

- オペレーティングシステムユーザー pem は、PEM サーバーのインストール中に作成されます。PEM アプリケーションデータとセッションは、このユーザーのホームディレクトリに保存されます。

サポートされているロケール

現在、Postgres Enterprise Manager サーバーと Web インターフェイスは、English(US) en_US ロケールと言語区切り文字としてのピリオド. の使用をサポートしています。代替ロケールまたはピリオド以外の区切り文字を使用すると、エラーが発生する場合があります。

6.1.4.3 Installing Postgres Enterprise Manager server on Linux AArch64 (ARM64)

オペレーティングシステム固有のインストール手順は、対応するドキュメントに記載されています。

Red Hat Enterprise Linux RHEL とその派生

- *Installing Postgres Enterprise Manager server on RHEL 9 or OL 9 arm64*
- *Installing Postgres Enterprise Manager server on RHEL 9 or OL 9 arm64*

Debian とデリバティブ

- *Installing Postgres Enterprise Manager server on Debian 12 arm64*

Installing Postgres Enterprise Manager server on RHEL 9 or OL 9 arm64

1 つのサーバーに PEM をインストールするか、Web アプリケーションサーバーとバックエンドデータベースを 2 つの別のサーバーにインストールできます。PEM インストールのためにサーバーを準備する必要があります。

前提条件を満たし、次の手順で説明するインストール手順を完了した後、*Web* **サーバー** *NGINX* **の構成** PEM を行う必要があります。2 つのサーバーを使用している場合は、両方のサーバーに PEM をインストールして構成します。

前提条件

インストールプロセスを開始する前に

1. バックエンドデータベースとして使用する PEM の supported Postgres instance をインストールします。

このインスタンスは、PEM Web アプリケーションに使用するのと同じサーバーにインストールすることも、別のサーバーにインストールすることもできます。次の手順で詳細に説明するように構成されている場合、既存の Postgres インスタンスを使用することもできます。

2. `pg_hba.conf` ファイルを更新することにより、Postgres バックエンドデータベースで認証を構成します。

構成する前に、次の変更を手動で加えます。*SELinux* **構成を変更する** の実行中に追加の変更が必要です。

- PEM に必要な関係を作成するために、PEM 構成スクリプトは、パスワード認証を使用して選択したスーパーユーザーとして Postgres バックエンドデータベースに接続します。これには、選択したスーパーユーザーにパスワードを使用した認証を許可する必要があります。このユーザーは、構成スクリプトを実行する任意の場所から接続できる必要があります。実際、これは、バックエンドデータベースが存在するサーバーと PEM Web アプリケーションがインストールされるサーバーが異なる場合を意味します。
- 選択したスーパーユーザーがパスワード認証を使用して接続できるようにするには、次の行を `pg_hba.conf` に追加します `host all superusername 127.0.0.1/32 scram-sha-256` などの `md5` または `scram-sha-256` 認証を使用した `host` 接続を許可します。

> **注釈**
>
> EDB Postgres Advanced Server を使用している場合、Modifying the pg_hba.conf file を参照してください。

PostgreSQL を使用している場合、 Client Authentication を参照してください。 .. ::

2_blank_1_3. sslutils 拡張機能が Postgres サーバーにインストールされていることを確認します。

RHEL/AlmaLinux/Rocky Linux または SLES で PostgreSQL または EDB Postgres Extended Server を使用している場合は、 `hstore contrib` モジュールをインストールする必要があります。

- EDB Postgres Advanced Server を使用している場合、次のように `sslutils` 拡張機能をインストールできます。 <x> は、 EDB Postgres Advanced サーバーバージョンです。

```
sudo dnf install edb-as<x>-server-sslutils
```

- PostgreSQL を使用している場合、次のように `sslutils` 、および必要に応じて `hstore` モジュールをインストールできます。<x> は PostgreSQL バージョンです。

```
sudo dnf install sslutils_<x> postgresql<x>-contrib
```

- EDB Postgres Extended Server を使用している場合、次のように `sslutils` 、および必要に応じて `hstore` モジュールをインストールできます。<x> は EDB Postgres Extended Server バージョンです。

```
sudo dnf install edb-postgresextended<x>-sslutils edb-postgresextended<x>-contrib
```

4. ファイアウォールを使用している場合は、PEM Web アプリケーションが配置されるサーバーでポート 8443 へのアクセスを許可します。

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

firewall-cmd --reload
```

5. Postgres Enterprise Manager が依存するコンポーネントがすべてのサーバーで最新の状態であることを確認します。これは、以下に示すようにパッケージマネージャーを使用してシステム全体を更新することにより実行できます。個々のパッケージを更新する場合、依存関係の完全なリストが *Dependencies of the PEM Server and Agent on Linux* で提供されています。

```
sudo dnf upgrade
```

パッケージをインストールする

```
sudo dnf -y install edb-pem
```

初期構成

```
#  You can configure the PEM server using the following command:

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

詳細については、 *Configuring the PEM server on Linux* を参照してください。

注釈

- オペレーティングシステムユーザー pem は、PEM サーバーのインストール中に作成されます。PEM アプリケーションデータとセッションは、このユーザーのホームディレクトリに保存されます。

サポートされているロケール

現在、Postgres Enterprise Manager サーバーと Web インターフェイスは、English(US) en_US ロケールと言語区切り文字としてのピリオド. の使用をサポートしています。代替ロケールまたはピリオド以外の区切り文字を使用すると、エラーが発生する場合があります。

Installing Postgres Enterprise Manager server on Debian 12 arm64

1 つのサーバーに PEM をインストールするか、Web アプリケーションサーバーとバックエンドデータベースを 2 つの別のサーバーにインストールできます。PEM インストールのためにサーバーを準備する必要があります。

前提条件を満たし、次の手順で説明するインストール手順を完了した後、*Web* サーバー *NGINX* の構成 PEM を行う必要があります。2 つのサーバーを使用している場合は、両方のサーバーに PEM をインストールして構成します。

前提条件

インストールプロセスを開始する前に

1. バックエンドデータベースとして使用する PEM の supported Postgres instance をインストールします。

このインスタンスは、PEM Web アプリケーションに使用するのと同じサーバーにインストールすることも、別のサーバーにインストールすることもできます。次の手順で詳細に説明するように構成されている場合、既存の Postgres インスタンスを使用することもできます。

2. pg_hba.conf ファイルを更新することにより、Postgres バックエンドデータベースで認証を構成します。

構成する前に、次の変更を手動で加えます。*SELinux* 構成を変更する の実行中に追加の変更が必要です。

- PEM に必要な関係を作成するために、PEM 構成スクリプトは、パスワード認証を使用して選択したスーパーユーザーとして Postgres バックエンドデータベースに接続します。これには、選択したスーパーユーザーにパスワードを使用した認証を許可する必要があります。このユーザーは、構成スクリプトを実行する任意の場所から接続できる必要があります。実際、これは、バックエンドデータベー

スが存在するサーバーと PEM Web アプリケーションがインストールされるサーバーが異なる場合を意味します。

- 選択したスーパーユーザーがパスワード認証を使用して接続できるようにするには、次の行を `pg_hba.conf` に追加します `host all superusername 127.0.0.1/32 scram-sha-256` などの `md5` または `scram-sha-256` 認証を使用した `host` 接続を許可します。

> **注釈**
>
> EDB Postgres Advanced Server を使用している場合、Modifying the pg_hba.conf file を参照してください。

PostgreSQL を使用している場合、 Client Authentication を参照してください。 .. ::
2_blank_1_3. sslutils 拡張機能が Postgres サーバーにインストールされていることを確認します。

RHEL/AlmaLinux/Rocky Linux または SLES で PostgreSQL または EDB Postgres Extended Server を使用している場合は、 `hstore contrib` モジュールをインストールする必要があります。

- EDB Postgres Advanced Server を使用している場合、次のように `sslutils` 拡張機能をインストールできます。 <x> は、 EDB Postgres Advanced サーバーバージョンです。

```
sudo apt-get install edb-as<x>-server-sslutils
```

- PostgreSQL を使用している場合、次のように `sslutils` 、および必要に応じて `hstore` モジュールをインストールできます。<x> は PostgreSQL バージョンです。

```
sudo apt-get install postgresql-<x>-sslutils
```

- EDB Postgres Extended Server を使用している場合、次のように `sslutils` 、および必要に応じて `hstore` モジュールをインストールできます。<x> は EDB Postgres Extended Server バージョンです。

```
sudo apt-get install edb-postgresextended-sslutils-<x>
```

4. ファイアウォールを使用している場合は、PEM Web アプリケーションが配置されるサーバーでポート 8443 へのアクセスを許可します。

5. Postgres Enterprise Manager が依存するコンポーネントがすべてのサーバーで最新の状態であることを確認します。これは、以下に示すようにパッケージマネージャーを使用してシステム全体を更新することにより実行できます。個々のパッケージを更新する場合、依存関係の完全なリストが *Dependencies of the PEM Server and Agent on Linux* で提供されています。

```
sudo apt-get upgrade
```

パッケージをインストールする

```
sudo apt-get -y install edb-pem
```

初期構成

```
#  You can configure the PEM server using the following command:

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

詳細については、 *Configuring the PEM server on Linux* を参照してください。

注釈

- オペレーティングシステムユーザー pem は、PEM サーバーのインストール中に作成されます。PEM アプリケーションデータとセッションは、このユーザーのホームディレクトリに保存されます。

サポートされているロケール

現在、Postgres Enterprise Manager サーバーと Web インターフェイスは、English(US) en_US ロケールと言語区切り文字としてのピリオド. の使用をサポートしています。代替ロケールまたはピリオド以外の区切り文字を使用すると、エラーが発生する場合があります。

6.1.4.4 Installing Postgres Enterprise Manager server on Linux IBM Power (ppc64le)

オペレーティングシステム固有のインストール手順は、対応するドキュメントに記載されています。

Red Hat Enterprise LinuxRHEL

- *Installing Postgres Enterprise Manager server on RHEL 9 or OL 9 arm64*
- *Installing Postgres Enterprise Manager server on RHEL 8 ppc64le*

SUSE Linux Enterprise SLES

- *Installing Postgres Enterprise Manager server on SLES 15 ppc64le*

Installing Postgres Enterprise Manager server on RHEL 9 ppc64le

1 つのサーバーに PEM をインストールするか、Web アプリケーションサーバーとバックエンドデータベースを 2 つの別のサーバーにインストールできます。PEM インストールのためにサーバーを準備する必要があります。

前提条件を満たし、次の手順で説明するインストール手順を完了した後、*Web* サーバー *NGINX* の構成 PEM を行う必要があります。2 つのサーバーを使用している場合は、両方のサーバーに PEM をインストールして構成します。

前提条件

インストールプロセスを開始する前に

1. バックエンドデータベースとして使用する PEM の supported Postgres instance をインストールします。

このインスタンスは、PEM Web アプリケーションに使用するのと同じサーバーにインストールすることも、別のサーバーにインストールすることもできます。次の手順で詳細に説明するように構成されている場合、既存の Postgres インスタンスを使用することもできます。

2. `pg_hba.conf` ファイルを更新することにより、Postgres バックエンドデータベースで認証を構成します。

構成する前に、次の変更を手動で加えます。*SELinux* **構成を変更する** の実行中に追加の変更が必要です。

- PEM に必要な関係を作成するために、PEM 構成スクリプトは、パスワード認証を使用して選択したスーパーユーザーとして Postgres バックエンドデータベースに接続します。これには、選択したスーパーユーザーにパスワードを使用した認証を許可する必要があります。このユーザーは、構成スクリプトを実行する任意の場所から接続できる必要があります。実際、これは、バックエンドデータベースが存在するサーバーと PEM Web アプリケーションがインストールされるサーバーが異なる場合を意味します。
- 選択したスーパーユーザーがパスワード認証を使用して接続できるようにするには、次の行を `pg_hba.conf` に追加します `host all superusername 127.0.0.1/32 scram-sha-256` などの `md5` または `scram-sha-256` 認証を使用した `host` 接続を許可します。

> **注釈**
>
> EDB Postgres Advanced Server を使用している場合、Modifying the pg_hba.conf file を参照してください。

PostgreSQL を使用している場合、 Client Authentication を参照してください。 .. ::
2_blank_1_3. sslutils 拡張機能が Postgres サーバーにインストールされていることを確認します。

RHEL/AlmaLinux/Rocky Linux または SLES で PostgreSQL または EDB Postgres Extended Server を使用している場合は、 `hstore contrib` モジュールをインストールする必要があります。

- EDB Postgres Advanced Server を使用している場合、次のように `sslutils` 拡張機能をインストールできます。 <x> は、 EDB Postgres Advanced サーバーバージョンです。

```
sudo dnf install edb-as<x>-server-sslutils
```

- PostgreSQL を使用している場合、次のように `sslutils` 、および必要に応じて `hstore` モジュールをインストールできます。<x> は PostgreSQL バージョンです。

```
sudo dnf install sslutils_<x> postgresql<x>-contrib
```

- EDB Postgres Extended Server を使用している場合、次のように `sslutils` 、および必要に応じて `hstore` モジュールをインストールできます。<x> は EDB Postgres Extended Server バージョンです。

```
sudo dnf install edb-postgresextended<x>-sslutils edb-postgresextended<x>-contrib
```

4. ファイアウォールを使用している場合は、PEM Web アプリケーションが配置されるサーバーでポート 8443 へのアクセスを許可します。

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

firewall-cmd --reload
```

5. Postgres Enterprise Manager が依存するコンポーネントがすべてのサーバーで最新の状態であることを確認します。これは、以下に示すようにパッケージマネージャーを使用してシステム全体を更新することにより実行できます。個々のパッケージを更新する場合、依存関係の完全なリストが *Dependencies of the PEM Server and Agent on Linux* で提供されています。

```
sudo dnf upgrade
```

パッケージをインストールする

```
sudo dnf -y install edb-pem
```

初期構成

```
#  You can configure the PEM server using the following command:

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

詳細については、 *Configuring the PEM server on Linux* を参照してください。

注釈

- オペレーティングシステムユーザー pem は、PEM サーバーのインストール中に作成されます。PEM アプリケーションデータとセッションは、このユーザーのホームディレクトリに保存されます。

サポートされているロケール

現在、Postgres Enterprise Manager サーバーと Web インターフェイスは、English(US) en_US ロケールと言語区切り文字としてのピリオド. の使用をサポートしています。代替ロケールまたはピリオド以外の区切り文字を使用すると、エラーが発生する場合があります。

Installing Postgres Enterprise Manager server on RHEL 8 ppc64le

1 つのサーバーに PEM をインストールするか、Web アプリケーションサーバーとバックエンドデータベースを 2 つの別のサーバーにインストールできます。PEM インストールのためにサーバーを準備する必要があります。

前提条件を満たし、次の手順で説明するインストール手順を完了した後、*Web* サーバー *NGINX* の構成 PEM を行う必要があります。2 つのサーバーを使用している場合は、両方のサーバーに PEM をインストールして構成します。

前提条件

インストールプロセスを開始する前に

1. バックエンドデータベースとして使用する PEM の supported Postgres instance をインストールします。

このインスタンスは、PEM Web アプリケーションに使用するのと同じサーバーにインストールすることも、別のサーバーにインストールすることもできます。次の手順で詳細に説明するように構成されている場合、既存の Postgres インスタンスを使用することもできます。

2. `pg_hba.conf` ファイルを更新することにより、Postgres バックエンドデータベースで認証を構成します。

構成する前に、次の変更を手動で加えます。*SELinux* **構成を変更する** の実行中に追加の変更が必要です。

- PEM に必要な関係を作成するために、PEM 構成スクリプトは、パスワード認証を使用して選択したスーパーユーザーとして Postgres バックエンドデータベースに接続します。これには、選択したスーパーユーザーにパスワードを使用した認証を許可する必要があります。このユーザーは、構成スクリプトを実行する任意の場所から接続できる必要があります。実際、これは、バックエンドデータベースが存在するサーバーと PEM Web アプリケーションがインストールされるサーバーが異なる場合を意味します。
- 選択したスーパーユーザーがパスワード認証を使用して接続できるようにするには、次の行を `pg_hba.conf` に追加します `host all superusername 127.0.0.1/32 scram-sha-256` などの `md5` または `scram-sha-256` 認証を使用した `host` 接続を許可します。

> **注釈**
>
> EDB Postgres Advanced Server を使用している場合、Modifying the pg_hba.conf file を参照してください。

PostgreSQL を使用している場合、 Client Authentication を参照してください。 .. ::
2_blank_1_3. sslutils 拡張機能が Postgres サーバーにインストールされていることを確認します。

RHEL/AlmaLinux/Rocky Linux または SLES で PostgreSQL または EDB Postgres Extended Server を使用している場合は、 `hstore contrib` モジュールをインストールする必要があります。

- EDB Postgres Advanced Server を使用している場合、次のように `sslutils` 拡張機能をインストールできます。 <x> は、 EDB Postgres Advanced サーバーバージョンです。

```
sudo dnf install edb-as<x>-server-sslutils
```

- PostgreSQL を使用している場合、次のように `sslutils` 、および必要に応じて `hstore` モジュールをインストールできます。<x> は PostgreSQL バージョンです。

```
sudo dnf install sslutils_<x> postgresql<x>-contrib
```

- EDB Postgres Extended Server を使用している場合、次のように `sslutils` 、および必要に応じて `hstore` モジュールをインストールできます。<x> は EDB Postgres Extended Server バージョンです。

```
sudo dnf install edb-postgresextended<x>-sslutils edb-postgresextended<x>-contrib
```

4. ファイアウォールを使用している場合は、PEM Web アプリケーションが配置されるサーバーでポート 8443 へのアクセスを許可します。

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

firewall-cmd --reload
```

5. Postgres Enterprise Manager が依存するコンポーネントがすべてのサーバーで最新の状態であることを確認します。これは、以下に示すようにパッケージマネージャーを使用してシステム全体を更新することにより実行できます。個々のパッケージを更新する場合、依存関係の完全なリストが *Dependencies of the PEM Server and Agent on Linux* で提供されています。

```
sudo dnf upgrade
```

パッケージをインストールする

```
sudo dnf -y install edb-pem
```

初期構成

```
#  You can configure the PEM server using the following command:

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

詳細については、 *Configuring the PEM server on Linux* を参照してください。

注釈

- オペレーティングシステムユーザー pem は、PEM サーバーのインストール中に作成されます。PEM アプリケーションデータとセッションは、このユーザーのホームディレクトリに保存されます。

サポートされているロケール

現在、Postgres Enterprise Manager サーバーと Web インターフェイスは、English(US) en_US ロケールと言語区切り文字としてのピリオド. の使用をサポートしています。代替ロケールまたはピリオド以外の区切り文字を使用すると、エラーが発生する場合があります。

Installing Postgres Enterprise Manager server on SLES 15 ppc64le

注釈

Postgres Enterprise Manager 8.3 以降は SLES でサポートされています。

前提条件

インストールプロセスを開始する前に

1. バックエンドデータベースとして使用する PEM の supported Postgres instance をインストールします。

このインスタンスは、PEM Web アプリケーションに使用するのと同じサーバーにインストールすることも、別のサーバーにインストールすることもできます。次の手順で詳細に説明するように構成されている場合、既存の Postgres インスタンスを使用することもできます。

2. `pg_hba.conf` ファイルを更新することにより、Postgres バックエンドデータベースで認証を構成します。

構成する前に、次の変更を手動で加えます。*SELinux* **構成を変更する** の実行中に追加の変更が必要です。

- PEM に必要な関係を作成するために、PEM 構成スクリプトは、パスワード認証を使用して選択したスーパーユーザーとして Postgres バックエンドデータベースに接続します。これには、選択したスーパーユーザーにパスワードを使用した認証を許可する必要があります。このユーザーは、構成スクリプトを実行する任意の場所から接続できる必要があります。実際、これは、バックエンドデータベースが存在するサーバーと PEM Web アプリケーションがインストールされるサーバーが異なる場合を意味します。
- 選択したスーパーユーザーがパスワード認証を使用して接続できるようにするには、次の行を `pg_hba.conf` に追加します `host all superusername 127.0.0.1/32 scram-sha-256` などの `md5` または `scram-sha-256` 認証を使用した `host` 接続を許可します。

> **注釈**
>
> EDB Postgres Advanced Server を使用している場合、Modifying the pg_hba.conf file を参照してください。

PostgreSQL を使用している場合、 Client Authentication を参照してください。 .. ::
2_blank_1_3. sslutils 拡張機能が Postgres サーバーにインストールされていることを確認します。

RHEL/AlmaLinux/Rocky Linux または SLES で PostgreSQL または EDB Postgres Extended Server を使用している場合は、 `hstore contrib` モジュールをインストールする必要があります。

- EDB Postgres Advanced Server を使用している場合、次のように `sslutils` 拡張機能をインストールできます。 <x> は、 EDB Postgres Advanced サーバーバージョンです。

```
sudo zypper install edb-as<x>-server-sslutils
```

- PostgreSQL を使用している場合、次のように `sslutils` 、および必要に応じて `hstore` モジュールをインストールできます。<x> は PostgreSQL バージョンです。

```
sudo zypper install sslutils_<x> postgresql<x>-contrib
```

- EDB Postgres Extended Server を使用している場合、次のように `sslutils` 、および必要に応じて `hstore` モジュールをインストールできます。<x> は EDB Postgres Extended Server バージョンです。

```
sudo zypper install edb-postgresextended<x>-sslutils edb-postgresextended<x>-contrib
```

4. ファイアウォールを使用している場合は、PEM Web アプリケーションが配置されるサーバーでポート 8443 へのアクセスを許可します。

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

firewall-cmd --reload
```

5. Postgres Enterprise Manager が依存するコンポーネントがすべてのサーバーで最新の状態であることを確認します。これは、以下に示すようにパッケージマネージャーを使用してシステム全体を更新することにより実行できます。個々のパッケージを更新する場合、依存関係の完全なリストが *Dependencies of the PEM Server and Agent on Linux* で提供されています。

```
sudo zypper update
```

パッケージをインストールする

```
sudo zypper -n install edb-pem
```

初期構成

```
#  You can configure the PEM server using the following command:

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

詳細については、 *Configuring the PEM server on Linux* を参照してください。

注釈

- オペレーティングシステムユーザー pem は、PEM サーバーのインストール中に作成されます。PEM アプリケーションデータとセッションは、このユーザーのホームディレクトリに保存されます。

サポートされているロケール

現在、Postgres Enterprise Manager サーバーと Web インターフェイスは、English(US) en_US ロケールと言語区切り文字としてのピリオド. の使用をサポートしています。代替ロケールまたはピリオド以外の区切り文字を使用すると、エラーが発生する場合があります。

6.1.4.5 Installing Postgres Enterprise Manager server on Windows

Windows では、PEM サーバーグラフィカルインストーラーは、PEM サーバー、PEM エージェント、および選択したブラウザーで PEM Web インターフェイスに接続するために必要なソフトウェアをインストールおよび構成します。

各 PEM インストールの中心となるのは PEM サーバーです。実稼働環境では、サーバーは通常専用マシンであり、多数の Postgres サーバーまたは少数のビジーサーバーを監視します。

PEM サーバーバックエンドデータベースは、PostgreSQL または EDB Postgres Advanced Server データベースサーバーの EDB ディストリビューション、または別のソースからインストールされた既存の Postgres サーバーです。Postgres バックエンドデータベースサーバーはバージョン 13 以降である必要があり、PEM サーバーがリポジトリとして使用する pem という名前のデータベースが含まれています。

- 既存の Postgres サーバーを使用して PEM サーバーをホストする場合、PEM サーバーインストーラーは Postgres ホストに pem データベースを作成できます。既存のサーバーを使用する場合、ソフトウェア前提条件を手動で満たす必要があります。

既存の Postgres サーバーを使用して PEM サーバーバックエンドデータベースをホストする方法の詳細については、*Installing the PEM server on an existing Postgres servers* を参照してください。

- Postgres の既存のインストールを PEM サーバーホストとして使用したくない場合、PEM サーバーインストーラーは次のことを行うことができます。
  - PostgreSQL のインストール
  - サーバーホストのソフトウェア前提条件を満たす s
  - pem データベースを含むインスタンス PostgreSQL データベースクラスターを作成します。これは、最も単純な PEM サーバーのインストールオプションです
- PEM-HTTPD は、PEM サーバーインストーラーまたは StackBuilder ユーティリティを介して Postgres インストールで利用できるようになります。PEM-HTTPD がホストに既にインストールされている場合、PEM サーバーインストーラーは、必要に応じて既存のインストールを確認および更新します。PEM サーバーホストに既存の PEM-HTTPD インストールが含まれていない場合、PEM サーバーインストーラーはそれを追加します。
- PEM サーバーをインストールする前に、PostgreSQL と PEM-HTTPD を同じホストで実行するか、別のホストで実行するかを決定する必要があります。PostgreSQL データベースサーバーと PEM-HTTPD を別のホストで実行する場合は、PEM サーバーインストーラーを各ホストで 1 回ずつ、2 回実行する必要があります。Installing the PEM server and PEM-HTTPD on separate hosts を参照してください。
- 同じホストへの PEM サーバーと PEM-HTTPD のインストールの詳細については、Installing the PEM server and PEM-HTTPD on the same host を参照してください。
- スタンドアロン PEM エージェントのインストールと構成の詳細については、*Installing Postgres Enterprise Manager agent on Windows* を参照してください。
- 言語パックインストーラーには、 EDB Postgres Advanced Server および EDB PostgreSQL データベースインストーラーで使用できるサポートされている言語が含まれています。言語パックインストーラーを使用すると、サードパーティベンダーからサポートソフトウェアをインストールせずに、Perl、TCL / TK、および Python をインストールできます。言語パックのインストールと使用の詳細については、EDB Postgres language pack を参照してください。
- PEM エージェントのインストールまたは構成のトラブルシューティングについては、*Troubleshooting agent issues* を参照してください。

PEM サーバーインストーラーは、PEM Web インターフェイスを使用してサーバーにアクセスするために必要なソフトウェアもインストールします。Web インターフェイスには、サポートされているバージョンのブラウザーでアクセスできます。Web インターフェイスを使用して、次のことができます。

- 監視対象サーバーにあるオブジェクトに関する情報を確認する

- 監視対象サーバーにあるデータベースとデータベースオブジェクトの管理
- PEM サーバーによって収集された統計情報を確認します

> **注釈**
>
> SSL 証明書を使用している場合、すべての SSL 証明書がバックエンドデータベースサーバーのデータディレクトリにあることを確認します。証明書がデータディレクトリにない場合、PEM サーバーの構成中にデータディレクトリを検索するため、PEM サーバーの構成が失敗する可能性があります。

Installing the PEM server and PEM-HTTPD on the same host

最も簡単な PEM サーバーインストール構成は、PEM バックエンドデータベースサーバー PEM サーバーインストーラーでインストールされた PostgreSQL データベースでホストされ、同じホストに存在する PEM-HTTPD サービスで構成されます。この構成では、PEM サーバーインストーラーは、バックエンドホストがサービスを登録するための前提条件ソフトウェアを提供します Windows。

1. PEM サーバーインストーラーを起動します。Windows システムでは、インストーラーアイコンを右クリックし、 **Run as Administrator** を選択します。インストーラーには、 Welcome 画面が表示されます。 **Next** を選択します。
2. 適切なラジオボタンを選択して契約に同意する前に、ライセンス契約を確認してください。**Next** を選択します。
3. Installation Directory ダイアログボックスを使用して、PEM サーバーの場所を指定します。

- デフォルトでは、PEM サーバーは Windows の `C:\Program Files\edb\pem` にインストールされます。デフォルトの場所をそのまま使用するか、**インストールディレクトリ**ボタンを使用してダイアログボックスを開き、PEM サーバーをインストールするディレクトリを選択します。
- **Show advanced options** チェックボックスを使用して、 Advanced Options ダイアログボックスを開きますインストール中に。Postgres データベースサーバと PEM-HTTPD を別のホストにインストールする場合、または既存の Postgres サーバーインストールに PEM サーバーを存在させたい場合に、Advanced Options ダイアログボックスを使用します。
- インストーラーと PEM-HTTPD を使用して PostgreSQL サーバーパッケージをインストールするには同じホストで、**Show advanced options** をクリアし、 **Next** を選択します。

PEM サーバーインストーラーは、PEM-HTTPD、言語パック、および PostgreSQL 13 のインストール前チェックを実行します。インストーラーがこれらのパッケージを見つけられない場合、[dependency Missing] ダイアログボックスで通知します。

> **注釈**
>
> デフォルトでは、EDB Language Pack は `C:edblanguagepackv1` にインストールされます。

1. 依存関係が欠落している場合、PEM サーバーインストーラーは、各インストールウィザードを起動します。各パッケージは、ウィザードの画面上の指示に従ってください。

不足している依存関係をインストールすると、[Database Server Installation Details] ダイアログボックスが表示されてインストールプロセスが続行されます。 [Database Server Installation Details] ダイアログボッ

クスで提供される情報により、インストーラーは PostgreSQL サーバーに接続できます。

1. データベースのスーパーユーザーのユーザー名とパスワードを入力します。要求された情報を入力したら、[**Next**] を選択します。

2. Postgres データベーススーパーユーザーの名前とパスワードを入力すると、PEM エージェントを実行するユーザーアカウントのパスワードの入力を求められる場合があります。プロンプトが表示されたら、パスワードを入力し、**Next** を選択します。

3. Network Details ダイアログボックスを使用して、PEM エージェントがサーバー**クライアントサイド**アドレスに接続する CIDR スタイルのネットワークアドレスを指定します。

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レンジを指定できます。たとえば、アドレス `192.168.10.23` 、`192.168.10.76` 、および `192.168.10.184` を持つデータベースサーバーを監視する場合は、そのネットワーク内のホストとの接続を許可するように `192.168.10.0/24` と入力します。

指定されたアドレスがサーバーの `pg_hba.conf` ファイルに追加されます。PostgreSQL サーバー上の `pg_hba.conf` ファイルにエントリを手動で追加することにより、より多くのネットワークアドレスを指定できます。最初のエントリをテンプレートとして使用します。

ネットワークアドレスを追加したら、[**Next**] を選択します。

PEM サーバーインストーラーは、サーバーが存在するホストに PEM エージェントをインストールして、サーバーを監視し、アラート処理とガベージコレクションサービスを提供します。証明書は、**エージェント証明書パス**フィールドで指定された場所にもインストールされます。

1. 代替説明を入力するか、PEM エージェントの代替エージェント証明書パスを選択するか、デフォルトを受け入れます。**Next** を選択します。

2. PEM によって生成された SSL 証明書で使用される組織の詳細を入力する必要があります。

3. ウィザードで PEM サーバーをインストールする準備が整いました。 **Next** を選択して、インストールを続行します。

インストールプロセス中に、インストーラーはファイルをシステムにコピーし、PEM の実行に必要なデータベースと Web サービスをセットアップします。インストールが完了すると、確認が次のことを示します。- Web サービスが構成されました。 - Web サービスはポート 8443 でリッスンしています。 - `pem` データベースが作成および構成されました。

1. **OK** を選択します。

Installing the PEM server and PEM-HTTPD on separate hosts

PEM サーバーバックエンドデータベースと PEM-HTTPD に別のホストを使用するには

1. `pem` データベースを含む Postgres サーバーのホストで PEM サーバーインストーラーを起動します。インストール中に、[Advanced Options] ダイアログボックスで [**Server**] オプションを選択し、Postgres サーバーの接続情報を提供します。

2. PEM サーバーおよび `pem` データベースが存在する Postgres インストールの `pg_hba.conf` ファイルを変更し、PEM-HTTPD サーバーのホストからの接続を許可します。

3. PEM-HTTPD サーバーのホストで PEM サーバーインストーラーを起動し、[Installation Type] ダイアログボックスで [**Web Services**] オプションを選択します。

インストールを開始するには

1. Windows システムでは、PEM サーバーインストーラーを起動します。インストーラーを右クリックし、[**Run as Administrator**] を選択します。インストーラーには、 Welcome 画面が表示されます。**Next** を選択します。

2. 適切なラジオボタンを選択して契約に同意する前に、ライセンス契約を確認してください。**Next** を選択します。

3. Installation Directory ダイアログボックスのフィールドを使用して、PEM サーバーのディレクトリを指定し、インストール中に Advanced Options ダイアログボックスを開きます。

   - デフォルトでは、PEM サーバーは Windows の `C:\Program Files\edb\pem` にインストールされます。デフォルトの場所をそのまま使用するか、**インストールディレクトリ**フィールドを使用してブラウザダイアログボックスを開き、PEM サーバーをインストールするディレクトリを選択します。

   - 別のホストに PEM サーバーと PEM-HTTPD をインストールするには、詳細オプションを使用しますダイアログボックスで、インストールタイプ **Web Services** または **Database** を指定します。インストールプロセスに Advanced Options ダイアログボックスを含めるには、 **Show advanced options** を選択します。

   - **Next** を選択します。

4. Advanced Options ダイアログボックスを使用して、インストールするコンポーネントを指定します。

   - **Web Services and Database** を選択して、Postgres サーバーと PEM-HTTPD の両方が現在のホストに存在することを示します。**Web Services と Database** を選択した場合、PEM サーバーインストーラーでは、PEM-HTTPD インストールをチェックする前に PEM サーバーに使用する Postgres サーバーを指定できます。

   - PEM-HTTPD をインストールするには **Web Services** を選択します別のホストにある Postgres データベースサーバーを使用して PEM サーバーと `pem` データベースをホストします。

> **注釈**
>
> **Web Services**オプションを使用して PEM-HTTPD をインストールする前に、PEM サーバーと `pem` バックエンドデータベースのホストで PEM サーバーのインストールプロセスを完了する必要があります。 Advanced Options ダイアログボックスで**Server**を選択し、PEM サーバー上の `pg_hba.conf` ファイルの接続プロパティを変更します。

このオプションは、*Web* **サービスのインストール** で説明されているインストール手順を呼び出します。

- **Database** を選択して、既存の Postgres サーバーバージョン 13 以降を使用するか、PEM サーバーインストーラーで配布されるデータベースサーバーのみをインストールします。このオプションは、**データベースホストの指定** で説明されているインストール手順を呼び出します。

1. インストールオプションを選択したら、[**Next**] を選択します。

データベースホストの指定

1. Advanced Options ダイアログボックスで **Server** を選択して、PEM サーバーのバックエンドデータベース pem という名前が存在するホストの接続情報を指定します。 **Next** を選択します。
2. [Database Server Selection] ダイアログボックスのリストを使用して、PEM サーバーバックエンドデータベースのホストを選択します。
   - 現在のホストに存在する既存の Postgres インストールからホストを選択します。

> **注釈**
>
> *sslutils* パッケージをインストールに追加する必要がある場合があります。*2_blank_4_- **PostgreSQL** を選択して、PEM サーバーインストーラーで配布される Postgres サーバーをインストールします。 <x> は、PostgreSQL データベースサーバーのバージョンです。 PEM サーバーインストーラーにバンドルされた PostgreSQL のバージョンを使用することにした場合は、EnterpriseDB ワンクリックの PostgreSQL インストーラーが開き、インストールをガイドします。*1_blank_4_- **Other Database Server**を選択して、次の Postgres サーバーの接続情報を指定します。 EnterpriseDB からワンクリックのグラフィカルインストーラーを使用してインストールされていません。 *1_blank_4_- ** Next**を選択します。

PEM サーバーが既存の Postgres サーバーにある場合、[Database Server Installation Details] ダイアログボックスが開きます。

1. [Database Server Installation Details] ダイアログボックスで必要な情報は異なる場合があります。PEM サーバーインストーラーは、選択したインストールに関する見つからない情報のみを提供するように求めます。
   - **User** フィールドに Postgres データベースのスーパーユーザーの名前を指定します。
   - そのユーザーに関連付けられたパスワードを指定します **Password** フィールド.
   - **Next** を選択します。
2. プロンプトが表示されたら、PEM エージェントを実行するサービスアカウントのシステムパスワードを入力します。**Next** を選択します。
3. Network Details ダイアログボックスを使用して、PEM エージェントがサーバークライアントサイドアドレスに接続する CIDR スタイルのネットワークアドレスを指定します。指定されたアドレスがサーバーの `pg_hba.conf` ファイルに追加されます。

> **注釈**
>
> PostgreSQL サーバー上の `pg_hba.conf` ファイルにエントリを手動で追加することにより、より多くのネットワークアドレスを指定できます。

デフォルトローカルホスト指定を受け入れるか、 **ネットワークアドレス**範囲を指定します。 **Next** を選択します。

PEM サーバーインストーラーは、サーバーが存在するホストに PEM エージェントをインストールして、サーバーを監視し、アラート処理とガベージコレクションサービスを提供します。証明書は、**エージェン**

ト証明書パスフィールドで指定された場所にもインストールされます。

1. PEM エージェントの代替説明または代替エージェント証明書パスを入力するか、デフォルトを受け入れます。**Next** を選択します。

2. ウィザードで PEM サーバーをインストールする準備が整いました。 **Next** を選択して、インストールを続行します。

インストールプロセス中に、インストーラーはシステムにファイルをコピーし、PEM サーバーのバックエンドデータベースをセットアップします。確認は、pem データベースが作成および構成されたことを示します。

1. **OK** を選択します

PEM サーバーのインストールのデータベース部分が完了したら、別のホストで PEM サーバーを呼び出して、PEM-HTTPD をインストールまたはアップグレードできます。

Web サービスのインストール

Advanced Options ダイアログボックスで **Web Services** を選択して、現在のホストに PEM-HTTPD をインストールするか、既存の PEM-HTTPD インストールを更新します。

> **注釈**
>
> 現在のホストは、PEM バッキングデータベースのホストではない可能性があります。

このオプションを選択する前に、以下を行う必要があります。

- ホストシステムに PEM サーバーインストーラーをインストールします。この際、PEM サーバーのバッキングデータベースを指定します。
- PEM サーバーデータベースホスト上の pg_hba.conf ファイルを変更して、PEM-HTTPD ホストからの接続を許可します。
- データベースサーバーを再起動します。

1. **Web Services** を選択し、 **Next** を選択します。PEM サーバーインストーラーは、現在のホストで既存の PEM-HTTPD および言語パックのインストールを確認します。

2. インストーラーがコンポーネントを見つけられない場合、インストーラーは、1 つ以上の依存関係が欠落していることを通知します。**Next** を選択して言語パックをインストールします。言語パックをインストールすると、インストーラーは PEM-HTTPD セットアップウィザードを起動します。

3. PEM-HTTPD Setup ウィザードの画面上の指示に従います。ウィザードがインストールを完了したら、[**Finish**] を選択して [Database Server Installation Details] ダイアログボックスを開きます。

4. Database Server Installation Details ダイアログボックスを使用して、PEM サーバーインストールをホストしている Postgres インストールの接続情報を提供します。

    - **Host** フィールドに PEM サーバーホストの名前または IP アドレスを入力します。
    - ポートを入力します Postgres サーバーが **ポート**フィールドにリッスンしている番号。
    - **User** フィールドに Postgres データベーススーパーユーザーの名前を入力します。

- **Password* に Postgres スーパーユーザーに関連付けられたパスワードを入力します。*フィールド。

5. **Next** を選択します。

PEM サーバーのインストールが完了する前に、インストーラーはデータベースホストに接続します。インストールを続行するには、PEM データベースホスト上の `pg_hba.conf` ファイルを httpd サーバーのホストからの接続を受け入れるように構成し、ファイアウォールが接続を許可する必要があります。PEM サーバーインストーラーは、PEM サーバーのインストールを完了し、PEM-HTTPD サーバーのホストに存在する必要がある項目のみを追加します。

Installing the PEM server on an existing Postgres servers

既存の Postgres サーバーバージョン 13 以降を使用して、PEM サーバーと `pem` データベースをホストできます。Postgres インストーラーと前提条件ソフトウェア拡張機能は、EnterpriseDB website で無料で入手できます。

PEM サーバーインストール用に既存の Postgres サーバーを構成できます。

> **注釈**
>
> これらの指示はガイドラインです。 Postgres インストールの構成に必要な実際の手順は、Postgres サーバーの構成によって異なります。

グラフィカルインストール

PEM サーバーインストーラーを起動します。管理者権限を想定し、インストーラーが含まれるディレクトリに移動します。次に、インストーラーを呼び出します。

```
pem_server-10.<x>.<x>-<x>-<platform>
```

<x> は PEM のメジャーバージョンとマイナーバージョンであり、<platform> はプラットフォームです。

次に、インストールを開始します。

1. インストーラーには、 Welcome 画面が表示されます。 **Next** を選択します。
2. 適切なラジオボタンを選択して契約に同意する前に、ライセンス契約を確認してください。**Next** を選択します。
3. Installation Directory ダイアログボックスを使用して PEM サーバーの場所を指定し、Advanced Options ダイアログボックスにアクセスします。
   - **Installation Directory** フィールドを使用してブラウザダイアログを開き、PEM サーバーをインストールするディレクトリを選択します。
   - 既存のサーバーに PEM サーバーをインストールする場合は、 **Show advanced options** を選択して、インストールプロセスに Advanced Options ダイアログボックスを含めます。
   - **Next** を選択します。
4. Advanced Options ダイアログボックスを使用して、インストールタイプを指定します。Postgres サーバーと PEM-HTTPD サーバーの両方が現在のホストに存在する場合、

- **Web Services and Database** を選択します。このオプションは、既存の Postgres サーバーを使用して PEM サーバーをホストする場合、または PEM サーバーインストーラーを使用して PEM サーバーが存在する場所に Postgres サーバーをインストールする場合に有効です。

Web サービスとデータベースを選択した場合、PEM サーバーインストーラーは、PEM-HTTPD インストールの現在のホストを確認し、必要に応じて PEM-HTTPD をアップグレードまたはインストールします。

- **Web Services** PEM-HTTPD サーバーのみの場合現在のホストに存在します。このオプションを呼び出す方法の詳細については、Installing web services を参照してください。現在のホストに PEM サーバーのみをインストールし、`pem` バックエンドデータベースを作成する場合は、
- **データベース**。このオプションは、既存の Postgres サーバーを使用して PEM サーバーをホストしている場合、または PEM サーバーインストーラーを使用して PEM が存在する PostgreSQL サーバーをインストールしている場合に有効です。

インストールオプションを選択したら、[**Next**] を選択します。

1. [Database Server Selection] ダイアログボックスのリストを使用して、PEM サーバーのバックエンドデータベースを選択します。
   - Postgres ワンクリックインストーラーまたは EDB Postgres Advanced Server インストーラーを使用してインストールされた、現在のホスト上の Postgres サーバーの名前を選択します.
   - **PostgreSQL x (Packaged)** を選択して、PEM サーバーインストーラーでパッケージ化された PostgreSQL サーバーをインストールして使用します。**x** は、PostgreSQL データベースサーバーのバージョンです。
   - 使用する **Other Database Server** を選択します EnterpriseDB インストーラー以外のソースからインストールされた Postgres データベース rpm やソースからビルドなど。

> **注釈**
>
> 選択したデータベースサーバーには、 *sslutils* contrib モジュールのインストールと登録済みサービス Windows が含まれている必要があります。

Advanced Options ダイアログボックスで **Web Services and Database** を選択した場合、インストールウィザードは、現在のホストで既存の PEM-HTTPD インストールを確認し、必要に応じてサービスをアップグレードまたはインストールします。

Advanced Options ダイアログボックスで **Server** を選択した場合、Database Server Installation Details ダイアログボックスが開きます。

2. [Database Server Installation Details] ダイアログボックスのフィールドを使用して、PEM サーバーをホストする Postgres サーバーへの接続を説明します。
   - **User** フィールドにデータベースのスーパーユーザーの名前を入力します。
   - 関連付けられたパスワードを入力します **Password** フィールドでスーパーユーザーと。

Next を選択します。

3. 実行する PEM エージェントサービスの管理者パスワードを入力します。**Next** を選択します。

4. Network Details ダイアログボックスを使用して、サーバーに接続する PEM エージェントの CIDR スタイルのネットワークアドレス `client-side` アドレスを指定します。指定されたアドレスがサーバーの `pg_hba.conf file.` に追加されます

PostgreSQL サーバー上の `pg_hba.conf` ファイルにエントリを手動で追加することにより、より多くのネットワークアドレスを指定できます。最初のエントリをテンプレートとして使用します。

ネットワークアドレスを追加したら、[**Next**] を選択します。

PEM サーバーインストーラーは、サーバーが存在するホストに PEM エージェントをインストールして、サーバーを監視し、アラート処理とガベージコレクションサービスを提供します。証明書は、**エージェント証明書パス**フィールドで指定された場所にもインストールされます。

5. PEM エージェントの代替説明または代替エージェント証明書パスを入力するか、デフォルトを受け入れます。**Next** を選択します。

6. PEM によって生成された SSL 証明書で使用される組織の詳細を入力する必要があります。

7. ウィザードで PEM サーバーをインストールする準備が整いました。 **Next** を選択して、インストールを続行します。

インストールプロセス中に、インストーラーはシステムにファイルをコピーし、PEM サーバーのバックエンドデータベースをセットアップします。確認は、`pem` データベースが作成および構成されたことを示します。

8. **OK** を選択します。

別のホストにある PEM-HTTPD サービスを使用している場合、以下を行う必要があります。

- Postgres サーバーホスト上の `pg_hba.conf` ファイルを変更して、ホスト間の接続を許可します。
- PEM-HTTPD サーバーのホストで PEM サーバーインストーラーを起動します。PEM-HTTPD のインストールの詳細については、Installing Web Services を参照してください。

コマンドラインからのサーバーインストーラーの呼び出し

PEM サーバーと PEM エージェントインストーラーのコマンドラインオプションは、リソースまたはシステム構成が制限されているためにグラフィカルインストールが機能しない場合に機能を提供します。次のことができます。

- インストーラーを呼び出すときに`--mode unattended` オプションを含めて、ユーザー入力なしでインストールを実行します。

すべてのコマンドラインオプションがすべてのプラットフォームに適しているわけではありません。コマンドラインオプションの完全なリファレンスガイドについては、インストーラーを起動するときに`--help` オプションを使用します。

無人モードでの PEM サーバーインストーラーの呼び出し

インストーラーを呼び出すときにコマンドラインでインストール設定を指定することにより、PEM サーバーの無人インストールを実行できます。PEM サーバーをインストールするシステムは、インターネットにアクセスできること必要です。

PEM サーバーをインストールするには管理者権限が必要です。PEM サーバーインストーラーを呼び出す前に、次の依存関係をインストールする必要があります。

- PostgreSQL
- pem-httpd
- 言語パック

PEM サーバーインストーラーを使用して、PEM サーバーの依存関係を満たすことができます。インストーラーの場所に移動し、次のコマンドを使用して依存関係を抽出します。

```
pem-server-10.<x>.<x>-windows-x64.exe --extract-dependents C:\
```

この例では、ファイルは C:\ ディレクトリに抽出されます。ファイルを抽出した後、各プログラムをインストールする必要があります。ファイルこの例では C:\) が含まれるディレクトリに移動し、次のように入力します。

```
edb-languagepack-<version>-windows-x64.exe --mode unattended
pem-httpd-<version>-windows-x64.exe --mode unattended
postgresql-<version>-windows-x64.exe --mode unattended
```

次に、PEM サーバーインストーラーを呼び出すことができます。

```
pem-server-10.<x>.<x>-windows-x64.exe --mode unattended
- -existing-user <registered_edb_user> --existing-password
<edb_user_password> --pgport <port> --pguser postgres
- -agent_description pem-agent --systempassword <windows_password>
- -agent-crt-path C:\edb
- -certi_subject_country
- -certi_subject_state US
- -certi_subject_city MyCity
- -certi_subject_org MyOrg
- -certi_subject_org_unit MyUnit
- -certi_subject_common_name MyName
- -certi_subject_email admin@myorg.domain
```

そこで

- registered_edb_user は、登録された EnterpriseDB ユーザーの名前を指定します。登録するには、EDB website にアクセスしてください。
- edb_user_password は、EDB ユーザーアカウントに関連付けられたパスワードを指定します。
- port は、バッキング PostgreSQL データベースが使用するポートを指定します。デフォルトでは、PostgreSQL データベースはポート 5432 を使用します。
- cidr_address_range は、PEM サーバーのバッキングデータベースの pg_hba.conf ファイルに追加して、サーバーが監視するエージェントからの接続を許可するアドレスレンジを指定します。ネットワーク範囲たとえば 192.168.2.0/24 を指定して、同じネットワーク上にあるエージェントにサーバーアクセスを提供できます。

- windows_password は、Windows 管理者のアカウントに関連付けられたパスワードを指定します。

> **注釈**
>
> PostgreSQL インストーラーは、無人モードで呼び出されると、パスワードが `postgres` である `postgres` という名前のユーザーを作成します。

6.1.4.6 Creating an EDB repository on an isolated network

PEM をアップグレードするサーバーが EDB リポジトリに直接アクセスできない場合、PEM RPM パッケージのホストとして機能するローカルリポジトリを作成できます。これは、必要な手順の高レベルの概要です。独自のネットワークに合わせてプロセスを変更する必要がある場合があります。

ローカルリポジトリを作成して使用するには

1. インターネットにアクセスできるシステムで次のコマンドを使用して、PEM の依存関係をダウンロードします。

```
yum install yum-plugin-downloadonly
mkdir /<pem_dir>
yum install --downloadonly --downloaddir=/<pem_dir>/ edb-pem
mkdir /<epel_dir>
yum install --downloadonly --downloaddir=/<epel_dir>/ epel-release*
```

<pem_dir> および<epel_dir> は、RPM をダウンロードするために作成するローカルディレクトリです。

2. ディレクトリ/<pem_dir> および/<epel_dir> を、隔離されたネットワーク内のマシンにコピーします。

3. リポジトリを作成します。

```
createrepo /<pem_dir>
createrepo /<epel_dir>
```

4. 次のことを指定する接続情報を使用して、/etc/yum.repos.d/pem.repo という名前のリポジトリー構成ファイルを作成します。

```
[pemrepo]
name=PEM Repository
baseurl=file:///pem7/
enabled=1
gpgcheck=0
```

5. 次のことを指定する接続情報を使用して、 /etc/yum.repos.d/epel.repo という名前のリポジトリー構成ファイルを作成します。

```
[epelrepo]
name=epel Repository
```

(次のページに続く)

(前のページからの続き)

```
baseurl=file:///pem7/
enabled=1
gpgcheck=0
```

6. ローカルリポジトリの場所と接続情報を指定した後、dnf コマンドを使用して PEM サーバーをインストールまたはアップグレードできます。

PEM サーバーをインストールするには

```
dnf install edb-pem
```

PEM サーバーをアップグレードするには

```
dnf upgrade edb-pem
```

6.1.4.7 Configuring the PEM server on Linux

PEM サーバーの構成

PEM サーバーパッケージには、Linux プラットフォームインストールの構成プロセスの自動化に役立つスクリプト `configure-pem-server.sh` が含まれています。スクリプトは`/usr/edb/pem/bin` ディレクトリにインストールされます。スクリプトを呼び出すには、root として次のコマンドを実行します。

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

> **注釈**
>
> SSL 証明書を提供する場合、すべての証明書がバックエンドデータベースサーバーのデータディレクトリにあることを確認します。証明書がデータディレクトリにない場合、PEM サーバーの構成中にデータディレクトリを調べるため、PEM サーバーの configure スクリプトは失敗する場合があります。

スクリプトを呼び出すときに、構成プロパティを指定するコマンドラインオプションを含めることができます。スクリプトは、コマンドラインで省略した値の入力を求めます。受け入れられるオプションは次のとおりです。

Short Option	Long Option	Description
-acp	*--pemagent-certificate-path*	PEM エージェント証明書パスを定義します。デフォルトは `/root/.pem`です。

次のページに続く

表 6 – 前のページからの続き

Short Option	Long Option	Description
-arp	*--agent-register-param*	*pemworker --register-agent*、たとえば *--enable-smtp=true* `または--display-name=xxx`などに渡す追加パラメーター。複数のオプションはコンマで区切る必要があります。
-aro	*--agent-register-overrider*	デフォルトのエージェント構成値たとえば `agent_id=1`または`max_connections=3`をオーバーライドします。複数のオプションはコンマで区切る必要があります。
-au	*--use-agent-user*	PEM エージェントのユーザー名。このパラメーターを使用して、個々の番号が付けられたエージェントユーザーたとえば agent1、agent2 などを生成するデフォルト動作をオーバーライドします。
-bhn	*--asb-host-name*	監視対象サーバーの IP アドレスまたは完全修飾ドメイン名を指定します。指定しない場合、`--host`パラメーターの値を取得します。
-ci	*--cidr-address*	エージェントがサーバーに接続する CIDR 形式のネットワークアドレスレンジ。サーバーの `pg_hba.conf`ファイル`192.168.1.0/24`などに追加されます。デフォルトは`0.0.0.0/0`です。
-dbi	*--db-install-path*	データベースサーバーインストールのディレクトリたとえば、 EDB Postgres Advanced Server の場合は`/usr/edb/as17`、PostgreSQL の場合は `/usr/pgsql-17`です。

次のページに続く

表 6 – 前のページからの続き

Short Option	Long Option	Description
-ds	*--db-unitfile*	PEM データベースサーバーのユニットファイル名。EDB Postgres Advanced Server の場合、デフォルトのファイル名は `edb-as-17`です。PostgreSQL の場合、それは `postgresql-17`です。
-h	*--help*	PEM サーバーの構成中に使用可能なすべてのオプションをリストします。
-ho	*--host*	PEM データベースサーバーのホストアドレス。 *Platform Compatibility* を使用している場合、これは IP またはホスト名のコンマ区切りリストです。構文の詳細については、libpq documentation を参照してください。
-kdc	*--skip-db-certificate*	データベース SSL 証明書を作成または構成しません。データベースサーバーで SSL を既に構成し、独自の証明書を提供する場合にこれを使用します。
-kwc	*--skip-web-certificate*	Web サーバー証明書を作成しません。独自の Web サーバー SSL 証明書を提供する場合にこれを使用します。
-nhc	*--no-hba-change*	`pg_hba.conf`および `postgresql.conf`ファイルへの変更をスキップします。
-p	*--port*	PEM データベースサーバーのポート番号。 *Platform Compatibility* を使用している場合、これはポートのコンマ区切りリストです。構文の詳細については、libpq documentation を参照してください。
-ps	*--pemagent-servicename*	pemagent のサービス名。デフォルト値は `pemagent`です。

次のページに続く

表 6 – 前のページからの続き

Short Option	Long Option	Description
-scs	*--server-certificate-subject*	カスタム Web サーバー証明書のサブジェクトを提供します。フォーマットは `/C=CountryCode/ST=State/L=City/O=Company/C` です。 `C=CountryCode`を`Alpha-2`コードとして提供します。
-sp	*--superpassword*	PEM データベースサーバーのスーパーユーザーパスワード。この値は必須です。
-srp	*--server-register-param*	*pemworker --register-server*、たとえば *--efm-cluster-name=xxx* `または`--config-dir=path`などに渡す追加パラメーター。
-su	*--superuser*	PEM データベースサーバーのスーパーユーザー名。
-t	*--type*	インストールタイプ 構成が Web サービスとバックエンドデータベースの場合は `1`、Web サービスを構成する場合は`2`、またはバックエンドデータベースを構成する場合は`3`を指定します。 `3`を指定する場合、データベースはローカルホストに存在する必要があります。
-uac	*--use-agent-sslcert*	生成する代わりに使用する既存のエージェント SSL 証明書ファイルへのパス。
-uak	*--use-agent-sslkey*	生成する代わりに使用する既存のエージェント SSL キーファイルへのパス。
-ues	*--use-existing-sslutils*	既存の sslutils を使用してエージェント証明書を生成します。 *--skip-db-certificate* `または`--use-trusted-ssl-db-certificates`とともに使用する場合にのみ効果があります。
-un	*--uninstall-pem-server*	PEM サーバーをアンインストールします。

次のページに続く

表 6 – 前のページからの続き

Short Option	Long Option	Description
-utd	*--use-trusted-ssl-db-certificates*	*--skip-db-certificate* `および`--no-hba-change`オプションを有効にするエンタープライズレベルのフラグ。
	--replication-solution	HA 展開のレプリケーションソリューションを指定します。有効な値は `patroni`または `efm`です。これが設定されている場合、*--server-register-param* `**オプションを使用して、選択したレプリケーションソリューションに必要なオプションを設定する必要があります。例については、***:ref:`Installing Postgres Enterprise Manager in HA Patterns<Installing Postgres Enterprise Manager in HA Patterns>*** を参照してください。
	--replication-role	HA 展開のこのホストのレプリケーションロールを指定します。有効な値は `primary`または `standby`です。

注釈

--asb-host-name オプションは以下には適用されません。 - PostgreSQL ソケット Linux でソケットパスを指定する場合を除きます。/で始まる必要があります。 - *PEM_MONITORED_SERVER_PASSWORD* ソケットパスが提供される場合の必須だが未使用の値としてを除く - *--server-addr* (*--asb-host-name* が提供されない場合のデフォルトを除きます)

コマンドラインで構成プロパティを指定しない場合、スクリプトは値の入力を求めます。スクリプトを呼び出すときに、次から選択します

1. `Web Services and Database` - Web サーバーとデータベースの両方が PEM サーバーと同じホストにある場合に、このオプションを選択します。
2. `Web Services` - Web サーバーが PEM サーバーとは別のホストにある場合、このオプションを選択します。
3. `Database` - PEM サーバーが使用する PEM バックエンドデータベースを構成するには、このオプションを選択します。指定されたデータベースはローカルホストに存在する必要があります。

> **注釈**
>
> Web サーバーとバックエンドデータベース PEM サーバーが別のホストにある場合、最初にデータベースサーバーオプション 3 を構成し、次に Web サービスオプション 2 を構成します。スクリプトは、Web サービスの前にバックエンドデータベースが構成されている場合にのみ続行されます。

構成オプションを選択すると、スクリプトは構成プロパティの入力を求めます。スクリプトが完了すると、PEM サーバーに必要なオブジェクトを作成するか、必要な構成手順を実行します。スクリプトのヘルプを表示するには、次のコマンドを使用します。

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

Web サーバーの選択

PEM は、Web アプリケーションのサーバーとして NGINX と Apache HTTPD の両方をサポートしています。新規インストールはデフォルトで NGINX を使用します。Web サーバーは、`configure-pem-server.sh` スクリプトによってインストールおよび構成されます。

Web サーバーを選択するには、スクリプトを実行する前に `USE_NGINX` 環境変数を設定します。

- `USE_NGINX=1` — NGINX を使用しますデフォルト
- `USE_NGINX=0` — Apache HTTPD を使用する

例、Apache HTTPD を使用するには

```
USE_NGINX=0 /usr/edb/pem/bin/configure-pem-server.sh
```

構成後に`/usr/edb/pem/bin/switch-web-server.sh` を実行して Web サーバーを変更できます。

SELinux の構成

システムで SELinux が有効になっている場合 RHEL 同様のシステムのデフォルト場合、PEM Web アプリケーションがインストールされているサーバーで `configure-selinux.sh` を実行する必要があります。デフォルトを使用している場合でも、Web 選択した Web サーバーを指定する必要があります。NGINX の場合は `USE_NGINX=1` 、Apache HTTPD の場合は `USE_NGINX=0` 。

```
USE_NGINX=1 /usr/edb/pem/bin/configure-selinux.sh
```

Web サーバーと PEM バックエンドデータベースが個別にインストールされている場合の構成後の手順

バックエンドデータベースとは別のホストで Web アプリケーションサーバーを実行する場合、PEM が完全に動作する前に、いくつかの追加の手動手順を実行する必要があります。

バックエンドの Postgres データベースが、Web アプリケーションサーバーから PEM へのログインが許可されたユーザーからの接続を受け入れることを確認します。これを行うには、次のエントリを `pg_hba.conf` に追加します。

```
host pem +pem_user <web_app_ip>/32 md5
```

<web_app_ip> は、Web アプリケーションサーバーの IP アドレスです。

さらに、Web アプリケーションサーバーの IP アドレスが、スクリプトの実行時に指定されたネットワークアドレス範囲内にない場合、2 つのエントリを追加して、このサーバーの PEM エージェントが接続できるようにする必要があります。

```
host pem +pem_agent <web_app_ip>/32 md5
host pem +pem_agent <web_app_ip>/32 cert
```

<web_app_ip> は、Web アプリケーションサーバーの IP アドレスです。

PEM アプリケーションへのアクセス

PEM サーバーを構成すると、ブラウザーで PEM Web インターフェイスにアクセスできます。次の場所に移動します

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

デフォルトでは、Web サービスはポート 8443 でリッスンします。ポートを変更するには、*Changing the default port* を参照してください。

6.1.4.8 Installing Postgres Enterprise Manager in HA Patterns

高可用性 PEM を展開する前に、High Availability Patterns for PEM Deployment の高レベルのガイダンスをよく理解してください。

PEM 10.4 以降の場合、HA PEM を展開するための推奨プロセスは次のとおりです。

- サポートされているクラスターマネージャーを使用して HA Postgres クラスターを展開します。
- 必要な SSL ファイルと HBA ルールを追加することにより、PEM インストールのクラスターを準備します。
- プライマリ、次にスタンバイに PEM をインストールおよび構成します。
- 別の Web サーバーを使用する場合、Web サーバーに PEM を展開します。

10. 4 より前のバージョンでは、configure-pem-server.sh スクリプトの制限のため、プロセスはより複雑でした。したがって、 HA PEM の新しい展開は最新バージョンで実行することを強くお勧めします。ただし、configure-pem-server.sh への変更には下位互換性があるため、以前のプロセスは調整なしで実行されます。

以下のページでは、上記の手順に従って、いくつかの HA PEM アーキテクチャの例の詳細な手順を提供します。これらは、考えられるすべてのパターンの完全なガイドであることを目的としたものではありません。

- *Deploying HA PEM with colocated web application and backend using EDB Failover Manager and Virtual IPs*

- *Deploying HA PEM with separate web application and backend using EDB Failover Manager and Virtual IPs*
- *Deploying HA PEM with separate web application and backend using Patroni and multi-host strings*

さらに、 HA PEM 展開の特定の要素への次のガイドが提供されます。

- *Storing user settings in the backend, accessible to all web application instances*

HA PEM using the C1 architecture with EFM and a Virtual IP

このページでは、リファレンスアーキテクチャ C1 に従って高可用性 HAPEM 展開をインストールおよび構成するための詳細な手順を提供します。

この例では、クラスター管理に EDB Failover ManagerEFM 5.2 を使用し、PEM バックエンドデータベースとして EDB Postgres Advanced Server 17 を使用し、RHEL のようなシステムでプライマリにトラフィックをルーティングするためのメカニズムとして仮想 IPVIP を使用します。他のオプションを理解するには、*High availability patterns for PEM deployment* を参照してください。

また、*Deciding on an SSL model* を参照して、インストールを開始する前に SSL 証明書とキーを提供する方法を決定します。

- 172.16.161.200-PEM プライマリ
- 172.16.161.201-PEM スタンバイ 1
- 172.16.161.202-PEM スタンバイ 2
- 172.16.161.203 - EFM Witness ノード
- 172.16.161.245-PEM VIP エージェントとユーザーが接続するために使用します

以下は VIP アドレスを使用する必要があります。

- 監視対象データベースサーバーの PEM エージェントバインディング
- PEM Web クライアントへのアクセス
- ウェブサーバーサービスへのアクセス

EFM クラスターの展開

PEM をインストールする前に、ホストで動作する EFM クラスターが必要です。このセクションでは、このようなクラスターを展開する手順を提供します。EFM クラスターをプロビジョニングする独自のプロセスがある場合、これらの手順に従う必要はありません。ただし、sslutils を使用する予定がある場合は、続行する前にクラスターにインストールされていることを確認する必要があります。

初期パッケージのインストールと Postgres 構成

すべてのホストで次の手順を実行します。証人には Postgres は必要ありません。

1. 次のパッケージをインストールします。
    - EDB Postgres Advanced Server PEM サーバーのバックエンドデータベース
    - EDB Failover Manager

```
dnf -qy module disable postgresql
dnf -y install epel-release
dnf config-manager --set-enabled crb
dnf -y install edb-as17-server edb-efm52
```

1. Postgres データベースを初期化し、サービスを開始します。

```
PGSETUP_INITDB_OPTIONS="-E UTF-8" /usr/edb/as17/bin/edb-as-17-setup initdb
systemctl start edb-as-17
systemctl enable edb-as-17
```

1. ファイアウォールで次のポートを開きます。

   - EPAS の場合は 5444
   - EFM の場合は 7800

例

```
firewall-cmd --zone=public --add-port=5444/tcp --permanent
firewall-cmd --zone=public --add-port=7800/tcp --permanent
firewall-cmd --reload
```

オプション sslutils のインストール

PEM が認証局として機能し、クライアント証明書を自動的に発行する場合、*Regenerating the agent SSL certificates for PEM using sslutils* を構成する必要があります。独自の証明書を提供する予定がある場合は、この手順をスキップできます。

1. すべての非 Witness ホストに sslutils パッケージをインストールします。

```
dnf install edb-as17-server-sslutils
```

ストリーミングレプリケーションのプライマリノードを設定する

プライマリでのみこれらの手順を実行します。

1. レプリケーションロールを作成します。

```
CREATE ROLE repl REPLICATION LOGIN PASSWORD password;
```

選択したパスワードを入力します。

2. postgresql.conf ファイルで次を構成します。

```
wal_level = replica
max_wal_senders = 10
wal_keep_size = 500
max_replication_slots = 10
```

ストリーミングレプリケーションのパラメーターの構成の詳細については、PostgreSQL documentation を参照してください。

3. ホストベースの認証/var/lib/edb/as17/data/pg_hba.conf ファイルに次のエントリを追加して、レプリケーションユーザーがすべてのスタンバイから接続できるようにします。

```
host  replication  repl  172.16.161.200/32  scram-sha-256
host  replication  repl  172.16.161.201/32  scram-sha-256
host  replication  repl  172.16.161.202/32  scram-sha-256
```

4. EPAS サーバーを再起動します。

```
systemctl restart edb-as-17.service
```

.pgpass ファイルを追加する

すべての非監視ノードでこれらの手順を実行します。

1. enterprisedb ユーザーのホームディレクトリに.pgpass ファイルを作成し、次のコンテンツを追加します。 <password> は、前に作成したレプリケーションユーザーのパスワードに置き換えます。

```
172.16.161.200:5444:replication:repl:<password>
172.16.161.201:5444:replication:repl:<password>
172.16.161.202:5444:replication:repl:<password>
```

1. ファイルにアクセス許可を設定してアクセスを制限します。

```
chmod 600 ~/.pgpass
```

ストリーミングレプリケーションのスタンバイノードを設定する

pg_basebackup ユーティリティを使用して、スタンバイサーバーに PEM バックエンドデータベースサーバーのレプリカを作成します。

1. すべてのスタンバイノードで EPAS のサービスを停止します。

```
systemctl stop edb-as-17.service
```

1. すべてのスタンバイノードでデータベースサーバーのデータディレクトリを削除します。

```
su - enterprisedb

rm -rf /var/lib/edb/as17/data/*
```

1. pg_basebackup を使用して、各スタンバイノードでプライマリノードのバックアップを作成します。

```
su - enterprisedb -c "\
    /usr/edb/as17/bin/pg_basebackup \
    -h 172.16.161.200 \
```

(次のページに続く)

(前のページからの続き)

```
    -D /var/lib/edb/as17/data \
    -U repl \
    -v -P -Fp -R -p 5444\
    "
```

1. 各スタンバイノードで EPAS データベースサーバーを起動します。

```
systemctl start edb-as-17
```

フェールオーバーを管理するように EFM を設定する

EFM をセットアップするには、次の手順を実行します。

1. プライマリで、データベースユーザー efm を作成して、データベースサーバーに接続します。ユーザーに WAL ログ関連機能の実行権限とモニタリング権限を付与します。スーパーユーザーとして

```
CREATE ROLE efm LOGIN PASSWORD password;

-- Give privilege to efm user to connect to a database
GRANT CONNECT ON DATABASE edb TO efm;

-- Give privilege to efm user to do backup operations
GRANT EXECUTE ON FUNCTION pg_current_wal_lsn() TO efm;
GRANT EXECUTE ON FUNCTION pg_last_wal_replay_lsn() TO efm;
GRANT EXECUTE ON FUNCTION pg_wal_replay_resume() TO efm;
GRANT EXECUTE ON FUNCTION pg_wal_replay_pause() TO efm;
GRANT EXECUTE ON FUNCTION pg_reload_conf() TO efm;

-- Grant monitoring privilege to the efm user
GRANT pg_monitor TO efm;
```

この変更はスタンバイに複製されます。

1. プライマリで、pg_hba.conf にエントリを追加して、efm データベースユーザーがすべてのノードからデータベースサーバーに接続できるようにします。これらのエントリはスタンバイに複製されます。

```
host      edb     efm     172.16.161.200/32     scram-sha-256
host      edb     efm     172.16.161.201/32     scram-sha-256
host      edb     efm     172.16.161.202/32     scram-sha-256
host      edb     efm     172.16.161.203/32     scram-sha-256
```

1. すべてのデータベースサーバーの構成をリロードします。

```
SELECT pg_reload_conf();
```

4. すべてのノードで、サンプルファイル/etc/edb/efm-5.2/efm.nodes.in を使用して efm.nodes ファイルを作成し、EFM OS ユーザーに読み取り/書き込みアクセスを付与します。

```
cp /etc/edb/efm-5.2/efm.nodes.in /etc/edb/efm-5.2/efm.nodes
chown efm:efm /etc/edb/efm-5.2/efm.nodes
chmod 600 /etc/edb/efm-5.2/efm.nodes
```

5. スタンバイノードで、プライマリノードの IP アドレスと EFM ポートを/etc/edb/efm-5.2/efm.nodes ファイルに追加します。

6. すべてのノードで、サンプルファイル/etc/edb/efm-5.2/efm.properties.in を使用して efm.properties ファイルを作成します。すべてのユーザーに読み取りアクセスを付与します。

```
cp /etc/edb/efm-5.2/efm.properties.in /etc/edb/efm-5.2/efm.properties
chown efm:efm /etc/edb/efm-5.2/efm.properties
chmod a+r /etc/edb/efm-5.2/efm.properties
```

7. 任意のノードで、 efm ユーティリティを使用して efm データベースユーザーのパスワード上記の手順 1 で定義を暗号化し、次のステップの出力をメモします。

```
export EFMPASS=password
/usr/edb/efm-5.2/bin/efm encrypt efm --from-env
```

8. すべてのノードで、efm.properties ファイルの次のパラメーターを編集します。 <encrypted-password> を前のステップの出力に置き換えます。 <ip-addr> は、各ノードの IP アドレスに置き換えます。

> **注釈**
>
> ホストがインターネットに接続していない場合は、 *ping.server* の値を、ping に応答するネットワーク上でアクセス可能な信頼できるサーバーのアドレスに置き換えます。

EFM 構成の詳細については、 the documentation を参照してください。 .. ::

```ini
db.user=efm db.password.encrypted=<encrypted-password> db.port=5444 db.database=edb db.service.owner=enterprisedb db.service.name=edb-as-17 db.bin=/usr/edb/as17/bin db.data.dir=/var/lib/edb/as17/data jdbc.sslmode=require user.email=username@example.com from.email=node1@efm-pem notification.text.prefix=[PEM/EFM] bind.address=<ip-addr>:7800 is.witness=false encrypt.agent.messages=true stop.isolated.primary=true stop.failed.primary=true primary.shutdown.as.failure=false ping.server.ip=8.8.8.8

# VIP configuration virtual.ip=172.16.161.245 virtual.ip.interface=ens33 virtual.ip.prefix=24 virtual.ip.single=true check.vip.before.promotion=true
```

9. 監視ノードで、is.witness 構成パラメーターの値を true に設定します。

```
is.witness=true
```

10. プライマリノードで、EFM サービスを有効にして開始します。

```
systemctl enable edb-efm-5.2
systemctl start edb-efm-5.2
```

11. プライマリノードで、スタンバイがクラスターに参加することを許可します。

```
/usr/edb/efm-5.2/bin/efm allow-node  efm  172.16.161.201
/usr/edb/efm-5.2/bin/efm allow-node  efm  172.16.161.202
/usr/edb/efm-5.2/bin/efm allow-node  efm  172.16.161.203
```

12. スタンバイノードと EFM 監視ノードで EFM サービスを有効にして開始します。

```
systemctl enable edb-efm-5.2
systemctl start edb-efm-5.2
```

13. 任意のノードで次のコマンドを実行して、EFM クラスターのステータスを確認します。

```
/usr/edb/efm-5.2/bin/efm cluster-status efm
```

出力は次のようになります。

```
Cluster Status: efm
    Agent Type  Address              DB       VIP
    ----------------------------------------------------------------
    Primary     172.16.161.200      UP       172.16.161.245*
    Standby     172.16.161.201      UP       172.16.161.245
    Standby     172.16.161.202      UP       172.16.161.245
    Witness     172.16.161.203      N/A      172.16.161.245

Allowed node host list:
    172.16.161.200 172.16.161.201 172.16.161.202 172.16.161.203

Membership coordinator: 172.16.161.200

Standby priority host list:
    172.16.161.201 172.16.161.202

Promote Status:

    DB Type     Address              WAL Received LSN   WAL Replayed LSN   Info
    ---------------------------------------------------------------------------
    Primary     172.16.161.200                         0/F7A3808
    Standby     172.16.161.201      0/F7A3808          0/F7A3808
    Standby     172.16.161.202      0/F7A3808          0/F7A3808

    Standby database(s) in sync with primary. It is safe to promote.
```

このステータスは、EFM が正常にセットアップされ、PEM サーバーのフェールオーバーを管理している

ことを確認します。

フェールオーバーが発生した場合、スタンバイのいずれかがプライマリノードとして昇格し、PEM エージェントが新しいプライマリノードに接続します。上記の手順を使用して、障害が発生したプライマリノードを新しいスタンバイに置き換えることができます。

PEM インストール用の EFM クラスターの準備

HA モードで PEM をインストールする前に、クラスターが 2 つの要件を満たしていることを確認する必要があります。

- PEM エージェントから PEM バックエンドデータベースへの接続に cert 認証方法を使用できるように SSL が構成されます。
- HBA ルールが適用され、エージェントと Web アプリケーションインスタンスが PEM バックエンドに接続できます。

SSL ファイルの準備

Generating SSL files for HA PEM backend hosts の指示に従ってください。

EFM クラスターを変更して SSL および HBA ルールを追加する

各 PEM ノード witness を除くで次のアクションを実行します。

1. postgresql.conf を変更して SSL を有効にし、上記で作成した SSL ファイルへのパスを設定します。

```
ssl = on
ssl_ca_file = root.crt
ssl_cert_file = server.crt
ssl_key_file = server.key
```

CRL がある場合は、次も設定します。

```
ssl_crl_file = root.crl
```

1. pg_hba.conf を変更して、PEM に必要なルールを追加します。このブロックをファイルの先頭、既存のコンテンツの上に追加し、クラスターに一致するように IP アドレスを調整します。

```
# Allow users of the web interface to connect to pem and postgres databases
hostssl pem      +pem_user   127.0.0.1/32          scram-sha-256
hostssl postgres +pem_user   127.0.0.1/32          scram-sha-256

# Allow the PEM server agents to register and connect to send data to PEM server
hostssl pem      +pem_agent  127.0.0.1/32          cert
hostssl pem      +pem_admin  172.16.161.200/32     scram-sha-256
hostssl pem      +pem_agent  172.16.161.200/32     cert
hostssl pem      +pem_admin  172.16.161.201/32     scram-sha-256
hostssl pem      +pem_agent  172.16.161.201/32     cert
hostssl pem      +pem_admin  172.16.161.202/32     scram-sha-256
```

(次のページに続く)

(前のページからの続き)

```
hostssl pem      +pem_agent  172.16.161.202/32      cert

# Allow our superuser to connect to perform cluster configuration
hostssl pem,postgres      pemsuper   172.16.161.200/32    scram-sha-256
hostssl pem,postgres      pemsuper   172.16.161.201/32    scram-sha-256
hostssl pem,postgres      pemsuper   172.16.161.202/32    scram-sha-256

# Allow monitoring agents to connect to PEM
# Specify a subnet that includes all your monitored servers
# Duplicate these lines for multiple subnets
hostssl pem      +pem_user   0.0.0.0/0               scram-sha-256
hostssl pem      +pem_agent  0.0.0.0/0               cert
```

1. 共通エージェントユーザーを使用する場合、PEM のインストール中に最初のエージェントが接続できるように 1 行追加します。この行は、必要に応じて後で削除できます。

```
hostssl pem      common_agent_user  127.0.0.1/32          cert
```

1. 前のセクションで追加した HBA ルールレプリケーションと EFM 用を `host` ではなく `hostssl` に変更して、すべての接続に SSL を適用します。

クラスターを再起動します

> **注釈**
>
> デフォルトでは、データベースサービスが停止すると EFM はフェイルオーバーをトリガーします。このローリング再起動中に、Postgres を再起動する前に EFM エージェントを停止して、意図しないフェイルオーバーを防ぎます。 !!!

クラスターのローリング再起動を実行します。それぞれの非 Witness ホストで、スタンバイからプライマリで最後に、次のことを実行します。

1. EFM エージェントサービスを停止します

```
systemctl stop edb-efm-5.2
```

1. Postgres を再起動します

```
systemctl restart edb-as-17
```

1. EFM エージェントサービスを開始します

```
systemctl start edb-efm-5.2
```

エージェントユーザーと証明書の準備

独自の SSL 証明書を提供する場合は、次の手順に従って PEM 展開の準備をします。sslutils を使用している場合、このセクションをスキップしてください。

プライマリで、共通エージェントユーザーを作成します。コモンエージェントユーザーにスーパーユーザー権限を付与します。これは一時的にのみ必要であり、後で取り消されます。

```
CREATE USER common_agent_user WITH SUPERUSER;
```

このユーザーの証明書とキーペアを生成し、すべてのホストに保存します。

証明書を生成する方法に関するガイダンスについては、*Generating agent certificates manually* を参照してください。

PEM は、3 つのホストに展開されたエージェントに対して、一意の番号付き Postgres ユーザー agent1〜agent3 を自動的に生成します。

この設定をサポートするには、3 つの異なる証明書/キーペア特定のエージェントユーザーごとに 1 つを手動で生成し、対応する各ホストに 1 つの一意のペアをインストールする必要があります。

プライマリに PEM をインストールする

プライマリホストでのみ次の手順を実行します。

1. PEM パッケージをインストールします。

```
dnf install edb-pem --allowerasing
```

1. パスワードを使用してログインできるスーパーユーザーを作成します。これは、PEM のインストールに使用するユーザーです。また、PEM ホストで実行されているエージェントのエージェントサーバーバインディングユーザーにもなります。

```
su - enterprisedb -c "psql -c \"create role pemsuper login superuser password your-
↪password-here;\""
```

1. PEM サーバーを構成するには、データベースオプションオプション 3 を使用して構成スクリプトを実行し、ホストを仮想 IPVIP に設定します。

システムが高可用性 HA 用に正しくセットアップされていることを確認するには、次のフラグを含める必要があります。

- **``--replication-solution efm`` & ``--replication-role primary``** これらのフラグはスクリプトを強制的に HA モードに設定し、フェールオーバーマネージャーとの互換性を保証します。
- **``-srp --efm-cluster-name`` & ``-srp --efm-install-path``** これらのオプションは、PEM が自分自身のバックエンドクラスターを適切にモニタできることを保証するために使用されます。
- **``--skip-db-certificate``** これにより、HA 環境で証明書を手動で管理しているため、スクリプトが SSL を自動構成しようとするのを防ぎます。

sslutils を使用する場合、`--use-existing-sslutils` を含める必要があります。これにより、スクリプトはデータベース SSL セットアップをスキップしますが、sslutils 拡張機能を利用してエージェント証明書の

生成を処理します。

```
/usr/edb/pem/bin/configure-pem-server.sh -t 1 \
                                         -ho 172.16.161.245 \
                                         -p 5444 \
                                         -su pemsuper \
                                         --asb-host-name 127.0.0.1 \
                                         --replication-solution efm \
                                         -srp --efm-cluster-name=efm \
                                         -srp --efm-install-path=/usr/edb/efm-5.2/ \
                                         --replication-role primary \
                                         --skip-db-certificate \
                                         --use-existing-sslutils \
                                         --db-install-path /usr/edb/as17/ \
                                         --db-unitfile edb-as-17
```

共通エージェントユーザを使用する場合、ユーザ名と、そのユーザ用に生成された証明書とキーへのパスを提供する必要があります。

```
/usr/edb/pem/bin/configure-pem-server.sh -t 1 \
                                         -ho 172.16.161.245 \
                                         -p 5444 \
                                         -su pemsuper \
                                         --asb-host-name 127.0.0.1 \
                                         --replication-solution efm \
                                         -srp --efm-cluster-name=efm \
                                         -srp --efm-install-path=/usr/edb/efm-5.2/ \
                                         --replication-role primary \
                                         --db-unitfile edb-as-17 \
                                         --db-install-path /usr/edb/as17/ \
                                         --skip-db-certificate \
                                         --use-agent-user common_agent_user \
                                         --use-agent-sslcert /path/to/agent/cert \
                                         --use-agent-sslkey /path/to/agent/key
```

このコマンドを実行した後、共通エージェントユーザーに pem_agent ロールを付与し、スーパーユーザーステータスを取り消します。

```
GRANT pem_agent to common_agent_user;
ALTER USER common_agent_user WITH NOSUPERUSER;
```

+pem_agent ルールと一致するため、必要に応じて、この段階で共通エージェントユーザーに追加した HBA ルールを削除することもできます。

独自の証明書を使用する場合、 agent_id を 1 として指定し、対応する agent1 ユーザー用に生成された証明書とキーへのパスを指定する必要があります。

```
/usr/edb/pem/bin/configure-pem-server.sh -t 1 \
                                         -ho 172.16.161.245 \
                                         -p 5444 \
                                         -su pemsuper \
                                         --asb-host-name 127.0.0.1 \
                                         --replication-solution efm \
                                         -srp --efm-cluster-name=efm \
                                         -srp --efm-install-path=/usr/edb/efm-5.2/ \
                                         --replication-role primary \
                                         --db-unitfile edb-as-17 \
                                         --db-install-path /usr/edb/as17/ \
                                         --skip-db-certificate \
                                         --use-agent-sslcert /path/to/agent/cert \
                                         --use-agent-sslkey /path/to/agent/key \
                                         -aro agent_id=1
```

さまざまな追加の詳細の入力を求められます。構成オプションについては、Configuring the PEM server on Linux を参照してください。

1. オプションで、インスタンス間で PEM Web アプリケーションのユーザー設定を同期するには、*configure central storage of user preferences* 。この段階では、バックエンド構成のみを完了できます。後で Web アプリケーションを構成します。

スタンバイに PEM をインストールする

スタンバイホストのみで次の手順を実行します。

1. PEM パッケージをインストールします。

```
dnf install edb-pem --allowerasing
```

1. オプション 3 データベースと VIP をホストとして指定して、PEM 構成スクリプトを実行します。`--replication-solution efm` および`--replication-role standby` オプションを指定して、HA モードが使用されるようにする必要があります。

```
/usr/edb/pem/bin/configure-pem-server.sh -t 1 \
                                         -ho 172.16.161.245 \
                                         -p 5444 \
                                         -su pemsuper \
                                         --asb-host-name 127.0.0.1 \
                                         --replication-solution efm \
                                         -srp --efm-cluster-name=efm \
                                         -srp --efm-install-path=/usr/edb/efm-5.2/ \
                                         --replication-role standby \
                                         --db-unitfile edb-as-17 \
                                         --db-install-path /usr/edb/as17/
```

共通エージェントユーザを使用する場合、ユーザ名と、そのユーザ用に生成された証明書とキーへのパス

を提供する必要があります。

```
/usr/edb/pem/bin/configure-pem-server.sh -t 1 \
                                         -ho 172.16.161.245 \
                                         -p 5444 \
                                         -su pemsuper \
                                         --asb-host-name 127.0.0.1 \
                                         --replication-solution efm \
                                         -srp --efm-cluster-name=efm \
                                         -srp --efm-install-path=/usr/edb/efm-5.2/ \
                                         --replication-role standby \
                                         --db-unitfile edb-as-17 \
                                         --db-install-path /usr/edb/as17/ \
                                         --use-agent-user common_agent_user \
                                         --use-agent-sslcert /path/to/agent/cert \
                                         --use-agent-sslkey /path/to/agent/key
```

独自の証明書を使用する場合、agent_id を次の整数 N として指定し、証明書へのパスと、対応する agentN ユーザー用に生成されたキーを指定する必要があります。

```
/usr/edb/pem/bin/configure-pem-server.sh -t 1 \
                                         -ho 172.16.161.245 \
                                         -p 5444 \
                                         -su pemsuper \
                                         --asb-host-name 127.0.0.1 \
                                         --replication-solution efm \
                                         -srp --efm-cluster-name=efm \
                                         -srp --efm-install-path=/usr/edb/efm-5.2/ \
                                         --replication-role standby \
                                         --db-unitfile edb-as-17 \
                                         --db-install-path /usr/edb/as17/ \
                                         --use-agent-sslcert /path/to/agent/cert \
                                         --use-agent-sslkey /path/to/agent/key \
                                         -aro agent_id=2
```

さまざまな追加の詳細の入力を求められます。構成オプションについては、Configuring the PEM server on Linux を参照してください。

PEM Web アプリケーションの最終手順

すべての Web アプリケーションホストで次の手順を実行します。

1. ファイアウォールで次のポートを開きます。

- PEM サーバー HTTPS の 8443

例

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

1. configure-selinux.sh スクリプトを実行して、PEM の SELinux ポリシーを構成します。

```
USE_NGINX=1 /usr/edb/pem/bin/configure-selinux.sh
```

1. これで、前に作成した pemsuper スーパーユーザーを使用して、Web アプリケーションインスタンスにログインできます。
2. インスタンス間で PEM Web アプリケーションのユーザー設定を同期することを選択した場合、*configuring each web application instance to use the backend for user settings* によって今すぐセットアップを完了します。

HA PEM using the S1 architecture with EFM and a Virtual IP

このページでは、リファレンスアーキテクチャ S1 に従って高可用性 HAPEM 展開をインストールおよび構成するための詳細な手順を提供します。

この例では、クラスター管理に EDB Failover ManagerEFM 5.2 を使用し、PEM バックエンドデータベースとして EDB Postgres Advanced Server 17 を使用し、RHEL のようなシステムでプライマリにトラフィックをルーティングするためのメカニズムとして仮想 IPVIP を使用します。

他のオプションを理解するには、 *High availability patterns for PEM deployment* を参照してください。

また、*Deciding on an SSL model* を参照して、インストールを開始する前に SSL 証明書とキーを提供する方法を決定します。

- 172.16.161.200-PEM バックエンドプライマリ
- 172.16.161.201-PEM バックエンドスタンバイ 1
- 172.16.161.202-PEM バックエンドスタンバイ 2
- 172.16.161.203 - EFM バックエンド Witness ノード
- 172.16.161.211-PEM Web アプリケーション 1
- 172.16.161.212-PEM Web アプリケーション 2
- 172.16.161.213-PEM Web アプリケーション 3
- 172.16.161.245-PEM VIP エージェントとユーザーが接続するために使用します

EFM クラスターの展開

PEM をインストールする前に、バックエンドホストで動作する EFM クラスターが必要です。このセクションでは、このようなクラスターを展開する手順を提供します。EFM クラスターをプロビジョニングする独自のプロセスがある場合、これらの手順に従う必要はありません。ただし、sslutils を使用する予定がある場合は、続行する前にクラスターにインストールされていることを確認する必要があります。

初期パッケージのインストールと Postgres 構成

特に記載がない限り、すべてのバックエンドノードで次の手順を実行します。

1. 次のパッケージをインストールします。

    - EDB Postgres Advanced Server PEM サーバーのバックエンドデータベース
    - EDB Failover Manager

```
dnf -qy module disable postgresql
dnf -y install epel-release
dnf config-manager --set-enabled crb
dnf -y install edb-as17-server edb-efm52
```

1. Postgres データベースを初期化し、サービスを開始します。

```
PGSETUP_INITDB_OPTIONS="-E UTF-8" /usr/edb/as17/bin/edb-as-17-setup initdb
systemctl start edb-as-17
systemctl enable edb-as-17
```

1. ファイアウォールで次のポートを開きます。

    - EPAS の場合は 5444
    - EFM の場合は 7800

例

```
firewall-cmd --zone=public --add-port=5444/tcp --permanent
firewall-cmd --zone=public --add-port=7800/tcp --permanent
firewall-cmd --reload
```

オプション sslutils のインストール

PEM が認証局として機能し、クライアント証明書を自動的に発行する場合、*Regenerating the agent SSL certificates for PEM using sslutils* を構成する必要があります。独自の証明書を提供する予定がある場合は、この手順をスキップできます。

1. すべてのバックエンドホストに sslutils パッケージをインストールします。

```
dnf install edb-as17-server-sslutils
```

ストリーミングレプリケーションのプライマリノードを設定する

プライマリでのみこれらの手順を実行します。

1. レプリケーションロールを作成します。

```
CREATE ROLE repl REPLICATION LOGIN PASSWORD password;
```

選択したパスワードを入力します。

2. postgresql.conf ファイルで次を構成します。

```
wal_level = replica
max_wal_senders = 10
wal_keep_size = 500
max_replication_slots = 10
```

ストリーミングレプリケーションのパラメーターの構成の詳細については、PostgreSQL documentation を参照してください。

3. ホストベースの認証/var/lib/edb/as17/data/pg_hba.conf ファイルに次のエントリを追加して、レプリケーションユーザーがすべてのスタンバイから接続できるようにします。

```
host  replication  repl  172.16.161.200/32  scram-sha-256
host  replication  repl  172.16.161.201/32  scram-sha-256
host  replication  repl  172.16.161.202/32  scram-sha-256
```

4. EPAS サーバーを再起動します。

```
systemctl restart edb-as-17.service
```

.pgpass ファイルを追加する

すべての非 Witness バックエンドノードでこれらの手順を実行します。

1. enterprisedb ユーザーのホームディレクトリに.pgpass ファイルを作成し、次のコンテンツを追加します。 <password> は、前に作成したレプリケーションユーザーのパスワードに置き換えます。

```
172.16.161.200:5444:replication:repl:<password>
172.16.161.201:5444:replication:repl:<password>
172.16.161.202:5444:replication:repl:<password>
```

1. ファイルにアクセス許可を設定してアクセスを制限します

```
chmod 600 ~/.pgpass
```

ストリーミングレプリケーションのスタンバイノードを設定する

pg_basebackup ユーティリティを使用して、スタンバイサーバーに PEM バックエンドデータベースサーバーのレプリカを作成します。

1. すべてのスタンバイノードで EPAS のサービスを停止します。

```
systemctl stop edb-as-17.service
```

1. すべてのスタンバイノードでデータベースサーバーのデータディレクトリを削除します。

```
su - enterprisedb

rm -rf /var/lib/edb/as17/data/*
```

1. pg_basebackup を使用して、各スタンバイノードでプライマリノードのバックアップを作成します。

```
su - enterprisedb -c "\
    /usr/edb/as17/bin/pg_basebackup \
    -h 172.16.161.200 \
    -D /var/lib/edb/as17/data \
    -U repl \
    -v -P -Fp -R -p 5444\
    "
```

1. 各スタンバイノードで EPAS データベースサーバーを起動します。

```
systemctl start edb-as-17
```

フェールオーバーを管理するように EFM を設定する

EFM をセットアップするには、次の手順を実行します。

1. プライマリで、データベースユーザー efm を作成して、データベースサーバーに接続します。ユーザーに WAL ログ関連機能の実行権限とモニタリング権限を付与します。スーパーユーザーとして

```
CREATE ROLE efm LOGIN PASSWORD password;

-- Give privilege to efm user to connect to a database
GRANT CONNECT ON DATABASE edb TO efm;

-- Give privilege to efm user to do backup operations
GRANT EXECUTE ON FUNCTION pg_current_wal_lsn() TO efm;
GRANT EXECUTE ON FUNCTION pg_last_wal_replay_lsn() TO efm;
GRANT EXECUTE ON FUNCTION pg_wal_replay_resume() TO efm;
GRANT EXECUTE ON FUNCTION pg_wal_replay_pause() TO efm;
GRANT EXECUTE ON FUNCTION pg_reload_conf() TO efm;

-- Grant monitoring privilege to the efm user
GRANT pg_monitor TO efm;
```

この変更はスタンバイに複製されます。

1. プライマリで、pg_hba.conf にエントリを追加して、efm データベースユーザーがすべてのノードからデータベースサーバーに接続できるようにします。これらのエントリはスタンバイに複製されます。

```
host      edb     efm     172.16.161.200/32     scram-sha-256
host      edb     efm     172.16.161.201/32     scram-sha-256
host      edb     efm     172.16.161.202/32     scram-sha-256
host      edb     efm     172.16.161.203/32     scram-sha-256
```

1. すべてのデータベースサーバーの構成をリロードします。

```
SELECT pg_reload_conf();
```

4. すべてのノードで、サンプルファイル/etc/edb/efm-5.2/efm.nodes.in を使用して efm.nodes ファイルを作成し、EFM OS ユーザーに読み取り/書き込みアクセスを付与します。

```
cp /etc/edb/efm-5.2/efm.nodes.in /etc/edb/efm-5.2/efm.nodes
chown efm:efm /etc/edb/efm-5.2/efm.nodes
chmod 600 /etc/edb/efm-5.2/efm.nodes
```

5. スタンバイノードで、プライマリノードの IP アドレスと EFM ポートを/etc/edb/efm-5.2/efm.nodes ファイルに追加します。

6. すべてのノードで、サンプルファイル/etc/edb/efm-5.2/efm.properties.in を使用して efm.properties ファイルを作成します。すべてのユーザーに読み取りアクセスを付与します。

```
cp /etc/edb/efm-5.2/efm.properties.in /etc/edb/efm-5.2/efm.properties
chown efm:efm /etc/edb/efm-5.2/efm.properties
chmod a+r /etc/edb/efm-5.2/efm.properties
```

7. 任意のノードで、 efm ユーティリティを使用して efm データベースユーザーのパスワード上記の手順 1 で定義を暗号化し、次のステップの出力をメモします。

```
export EFMPASS=password
/usr/edb/efm-5.2/bin/efm encrypt efm --from-env
```

8. すべてのノードで、efm.properties ファイルの次のパラメーターを編集します。 <encrypted-password> を前のステップの出力に置き換えます。 <ip-addr> は、各ノードの IP アドレスに置き換えます。

> **注釈**
>
> ホストがインターネットに接続していない場合は、 *ping.server* の値を、ping に応答するネットワーク上でアクセス可能な信頼できるサーバーのアドレスに置き換えます。

EFM 構成の詳細については、 the documentation を参照してください。 .. ::

```ini
db.user=efm db.password.encrypted=<encrypted-password> db.port=5444 db.database=edb
db.service.owner=enterprisedb db.service.name=edb-as-17 db.bin=/usr/edb/as17/bin
db.data.dir=/var/lib/edb/as17/data jdbc.sslmode=require user.email=username@example.com
from.email=node1@efm-pem notification.text.prefix=[PEM/EFM] bind.address=<ip-addr>:7800
```

is.witness=false encrypt.agent.messages=true stop.isolated.primary=true stop.failed.primary=true primary.shutdown.as.failure=false ping.server.ip=8.8.8.8

VIP configuration virtual.ip=172.16.161.245 virtual.ip.interface=ens33 virtual.ip.prefix=24 virtual.ip.single=true check.vip.before.promotion=true

```

9. 監視ノードで、is.witness 構成パラメーターの値を true に設定します。

```
is.witness=true
```

10. プライマリノードで、EFM サービスを有効にして開始します。

```
systemctl enable edb-efm-5.2
systemctl start edb-efm-5.2
```

11. プライマリノードで、スタンバイがクラスターに参加することを許可します。

```
/usr/edb/efm-5.2/bin/efm allow-node  efm  172.16.161.201
/usr/edb/efm-5.2/bin/efm allow-node  efm  172.16.161.202
/usr/edb/efm-5.2/bin/efm allow-node  efm  172.16.161.203
```

12. スタンバイノードと EFM 監視ノードで EFM サービスを有効にして開始します。

```
systemctl enable edb-efm-5.2
systemctl start edb-efm-5.2
```

13. 任意のノードで次のコマンドを実行して、EFM クラスターのステータスを確認します。

```
/usr/edb/efm-5.2/bin/efm cluster-status efm
```

出力は次のようになります。

```
Cluster Status: efm
    Agent Type  Address              DB       VIP
    ------------------------------------------------------------------
    Primary     172.16.161.200      UP       172.16.161.245*
    Standby     172.16.161.201      UP       172.16.161.245
    Standby     172.16.161.202      UP       172.16.161.245
    Witness     172.16.161.203      N/A      172.16.161.245

Allowed node host list:
    172.16.161.200 172.16.161.201 172.16.161.202 172.16.161.203

Membership coordinator: 172.16.161.200

Standby priority host list:
    172.16.161.201 172.16.161.202
```

(次のページに続く)
```

(前のページからの続き)

```
Promote Status:

    DB Type     Address              WAL Received LSN   WAL Replayed LSN   Info
    --------------------------------------------------------------------------------
    Primary     172.16.161.200                          0/F7A3808
    Standby     172.16.161.201      0/F7A3808           0/F7A3808
    Standby     172.16.161.202      0/F7A3808           0/F7A3808

    Standby database(s) in sync with primary. It is safe to promote.
```

このステータスは、EFM が正常にセットアップされ、PEM サーバーのフェールオーバーを管理していることを確認します。

フェールオーバーが発生した場合、スタンバイのいずれかがプライマリノードとして昇格し、PEM エージェントが新しいプライマリノードに接続します。上記の手順を使用して、障害が発生したプライマリノードを新しいスタンバイに置き換えることができます。

PEM インストール用の EFM クラスターの準備

HA モードで PEM をインストールする前に、クラスターが 2 つの要件を満たしていることを確認する必要があります。

- PEM エージェントから PEM バックエンドデータベースへの接続に `cert` 認証方法を使用できるように SSL が構成されます。
- HBA ルールが適用され、エージェントと Web アプリケーションインスタンスが PEM バックエンドに接続できます。

SSL ファイルの準備

Generating SSL files for HA PEM backend hosts の指示に従ってください。

EFM クラスターを変更して SSL および HBA ルールを追加する

各 PEM バックエンドノード witness を除くで次のアクションを実行します。

1. `postgresql.conf` を変更して SSL を有効にし、上記で作成した SSL ファイルへのパスを設定します。

```
ssl = on
ssl_ca_file = root.crt
ssl_cert_file = server.crt
ssl_key_file = server.key
```

CRL がある場合は、次も設定します。

```
ssl_crl_file = root.crl
```

1. pg_hba.conf を変更して、PEM に必要なルールを追加します。このブロックをファイルの先頭、既存のコンテンツの上に追加し、クラスターに一致するように IP アドレスを調整します。

```
# Allow users of the web interface to connect to pem and postgres databases
hostssl pem      +pem_user   127.0.0.1/32           scram-sha-256
hostssl postgres +pem_user   127.0.0.1/32           scram-sha-256
hostssl pem      +pem_user   172.16.161.211/32      scram-sha-256
hostssl postgres +pem_user   172.16.161.211/32      scram-sha-256
hostssl pem      +pem_user   172.16.161.212/32      scram-sha-256
hostssl postgres +pem_user   172.16.161.212/32      scram-sha-256
hostssl pem      +pem_user   172.16.161.213/32      scram-sha-256
hostssl postgres +pem_user   172.16.161.213/32      scram-sha-256

# Allow the PEM backend server agents to register and connect to send data to PEM␣
↪server
hostssl pem      +pem_agent  127.0.0.1/32           cert
hostssl pem      +pem_admin  172.16.161.200/32      scram-sha-256
hostssl pem      +pem_agent  172.16.161.200/32      cert
hostssl pem      +pem_admin  172.16.161.201/32      scram-sha-256
hostssl pem      +pem_agent  172.16.161.201/32      cert
hostssl pem      +pem_admin  172.16.161.202/32      scram-sha-256
hostssl pem      +pem_agent  172.16.161.202/32      cert

# Allow the PEM web app server agents to register and connect to send data to PEM␣
↪server
hostssl pem      +pem_admin   172.16.161.211/32    scram-sha-256
hostssl pem      +pem_agent   172.16.161.211/32    cert
hostssl pem      +pem_admin   172.16.161.212/32    scram-sha-256
hostssl pem      +pem_agent   172.16.161.212/32    cert
hostssl pem      +pem_admin   172.16.161.213/32    scram-sha-256
hostssl pem      +pem_agent   172.16.161.213/32    cert

# Allow our superuser to connect to perform cluster configuration
hostssl pem,postgres      pemsuper   172.16.161.200/32    scram-sha-256
hostssl pem,postgres      pemsuper   172.16.161.201/32    scram-sha-256
hostssl pem,postgres      pemsuper   172.16.161.202/32    scram-sha-256
hostssl pem,postgres      pemsuper   172.16.161.211/32    scram-sha-256
hostssl pem,postgres      pemsuper   172.16.161.212/32    scram-sha-256
hostssl pem,postgres      pemsuper   172.16.161.213/32    scram-sha-256

# Allow monitoring agents to connect to PEM
# Specify a subnet that includes all your monitored servers
# Duplicate these lines for multiple subnets
hostssl pem      +pem_user   0.0.0.0/0              scram-sha-256
hostssl pem      +pem_agent  0.0.0.0/0              cert
```

1. 共通エージェントユーザーを使用する場合、PEM のインストール中に最初のエージェントが接続で

きるように 1 行追加します。この行は、必要に応じて後で削除できます。

```
hostssl pem      common_agent_user  127.0.0.1/32          cert
```

1. 前のセクションで追加した HBA ルールレプリケーションと EFM 用を host ではなく hostssl に変更して、すべての接続に SSL を適用します。

クラスターを再起動します

> **注釈**
>
> デフォルトでは、データベースサービスが停止すると EFM はフェイルオーバーをトリガーします。このローリング再起動中に、Postgres を再起動する前に EFM エージェントを停止して、意図しないフェイルオーバーを防ぎます。 !!!

クラスターのローリング再起動を実行します。各 PEM バックエンドホストで、スタンバイからプライマリで終わり、次のことを実行します。

1. EFM エージェントサービスを停止します。

```
systemctl stop edb-efm-5.2
```

1. Postgres を再起動します。

```
systemctl restart edb-as-17
```

1. EFM エージェントサービスを開始します。

```
systemctl start edb-efm-5.2
```

エージェントユーザーと証明書の準備

独自の SSL 証明書を使用している場合は、次の手順に従って PEM 展開の準備をします。sslutils を使用している場合、このセクションをスキップしてください。

プライマリで、共通エージェントユーザーを作成します。コモンエージェントユーザーにスーパーユーザー権限を付与します。これは一時的にのみ必要であり、後で取り消されます。

```
CREATE USER common_agent_user WITH SUPERUSER;
```

このユーザーの証明書とキーペアを生成し、すべてのバックエンドおよび Web アプリケーションホストに保存します。

証明書を生成する方法に関するガイダンスについては、*Generating agent certificates manually* を参照してください。

PEM は、3 つのバックエンドと 3 つのフロントエンドホストにわたる展開に対して、一意の番号付けの Postgres ユーザー agent1〜agent6 を自動的に生成します。

この構成をサポートするには、6 つの異なる証明書/キーペア特定のエージェントユーザーごとに 1 つを手動で生成し、各ホストが独自の一意のペアを持つようにそれらを配布する必要があります。

プライマリに PEM をインストールする

プライマリホストでのみ次の手順を実行します。

1. PEM パッケージをインストールします。

```
dnf install edb-pem --allowerasing
```

1. パスワードを使用してログインできるスーパーユーザーを作成します。これは、PEM のインストールに使用するユーザーです。また、PEM バックエンドホストで実行されているエージェントのエージェントサーバーバインディングユーザーにもなります。

```
su - enterprisedb -c "psql -c \"create role pemsuper login superuser password your-
↪password-here;\""
```

1. PEM サーバーを構成するには、データベースオプションオプション 3 を使用して構成スクリプトを実行し、ホストを仮想 IPVIP に設定します。

システムが高可用性 HA 用に正しくセットアップされていることを確認するには、次のフラグを含める必要があります。

- **``--replication-solution efm`` & ``--replication-role primary``** これらのフラグはスクリプトを強制的に HA モードに設定し、フェールオーバーマネージャーとの互換性を保証します。
- **``-srp --efm-cluster-name`` & ``-srp --efm-install-path``** これらのオプションは、PEM が自分自身のバックエンドクラスターを適切にモニタできることを保証するために使用されます。
- **``--skip-db-certificate``** これにより、HA 環境で証明書を手動で管理しているため、スクリプトが SSL を自動構成しようとするのを防ぎます。

sslutils を使用する場合、`--use-existing-sslutils` を含める必要があります。これにより、スクリプトはデータベース SSL セットアップをスキップしますが、sslutils 拡張機能を利用してエージェント証明書の生成を処理します。

```
/usr/edb/pem/bin/configure-pem-server.sh -t 3 \
                                         -ho 172.16.161.245 \
                                         -p 5444 \
                                         -su pemsuper \
                                         --asb-host-name 127.0.0.1 \
                                         --replication-solution efm \
                                         -srp --efm-cluster-name=efm \
                                         -srp --efm-install-path=/usr/edb/efm-5.2/ \
                                         --replication-role primary \
                                         --skip-db-certificate \
                                         --use-existing-sslutils \
                                         --db-install-path /usr/edb/as17/ \
                                         --db-unitfile edb-as-17
```

共通エージェントユーザを使用する場合、ユーザ名と、そのユーザ用に生成された証明書とキーへのパスを提供する必要があります。

```
/usr/edb/pem/bin/configure-pem-server.sh -t 3 \
                                         -ho 172.16.161.245 \
                                         -p 5444 \
                                         -su pemsuper \
                                         --asb-host-name 127.0.0.1 \
                                         --replication-solution efm \
                                         -srp --efm-cluster-name=efm \
                                         -srp --efm-install-path=/usr/edb/efm-5.2/ \
                                         --replication-role primary \
                                         --db-unitfile edb-as-17 \
                                         --db-install-path /usr/edb/as17/ \
                                         --skip-db-certificate \
                                         --use-agent-user common_agent_user \
                                         --use-agent-sslcert /path/to/agent/cert \
                                         --use-agent-sslkey /path/to/agent/key
```

このコマンドを実行した後、共通エージェントユーザーに pem_agent ロールを付与し、スーパーユーザーステータスを取り消します。

```
GRANT pem_agent to common_agent_user;
ALTER USER common_agent_user WITH NOSUPERUSER;
```

+pem_agent ルールと一致するため、必要に応じて、この段階で共通エージェントユーザーに追加した HBA ルールを削除することもできます。

独自の証明書を使用する場合、 agent_id を 1 として指定し、対応する agent1 ユーザー用に生成された証明書とキーへのパスを指定する必要があります。

```
/usr/edb/pem/bin/configure-pem-server.sh -t 3 \
                                         -ho 172.16.161.245 \
                                         -p 5444 \
                                         -su pemsuper \
                                         --asb-host-name 127.0.0.1 \
                                         --replication-solution efm \
                                         -srp --efm-cluster-name=efm \
                                         -srp --efm-install-path=/usr/edb/efm-5.2/ \
                                         --replication-role primary \
                                         --db-unitfile edb-as-17 \
                                         --db-install-path /usr/edb/as17/ \
                                         --skip-db-certificate \
                                         --use-agent-sslcert /path/to/agent/cert \
                                         --use-agent-sslkey /path/to/agent/key \
                                         -aro agent_id=1
```

さまざまな追加の詳細の入力を求められます。構成オプションについては、Configuring the PEM server on

Linux を参照してください。

1. オプションで、インスタンス間で PEM Web アプリケーションのユーザー設定を同期するには、*configure central storage of user preferences* 。この段階では、バックエンド構成のみを完了できます。後で Web アプリケーションを構成します。

スタンバイに PEM をインストールする

スタンバイホストのみで次の手順を実行します。

1. PEM パッケージをインストールします。

```
dnf install edb-pem --allowerasing
```

1. PEM サーバーを構成するには、データベースオプションオプション 3 を使用して構成スクリプトを実行し、ホストを仮想 IPVIP に設定します。

高可用性 HA モードを正しく有効にするには、次の特定のフラグを含める必要があります。

- ``**--replication-solution efm**`` 構成を Enterprise Failover ManagerEFM と統合します。
- ``**--replication-role standby**`` 内でのこのノードの初期ロールを指定します HA クラスター。

```
/usr/edb/pem/bin/configure-pem-server.sh -t 3 \
                                         -ho 172.16.161.245 \
                                         -p 5444 \
                                         -su pemsuper \
                                         --asb-host-name 127.0.0.1 \
                                         --replication-solution efm \
                                         -srp --efm-cluster-name=efm \
                                         -srp --efm-install-path=/usr/edb/efm-5.2/ \
                                         --replication-role standby \
                                         --db-unitfile edb-as-17 \
                                         --db-install-path /usr/edb/as17/
```

共通エージェントユーザを使用する場合、ユーザ名と、そのユーザ用に生成された証明書とキーへのパスを提供する必要があります。

```
/usr/edb/pem/bin/configure-pem-server.sh -t 3 \
                                         -ho 172.16.161.245 \
                                         -p 5444 \
                                         -su pemsuper \
                                         --asb-host-name 127.0.0.1 \
                                         --replication-solution efm \
                                         -srp --efm-cluster-name=efm \
                                         -srp --efm-install-path=/usr/edb/efm-5.2/ \
                                         --replication-role standby \
                                         --db-unitfile edb-as-17 \
                                         --db-install-path /usr/edb/as17/ \
                                         --use-agent-user common_agent_user \
```

(次のページに続く)

(前のページからの続き)

```
                                        --use-agent-sslcert /path/to/agent/cert \
                                        --use-agent-sslkey /path/to/agent/key
```

独自の証明書を使用する場合、agent_id を次の整数 N として指定し、証明書へのパスと、対応する agentN ユーザー用に生成されたキーを指定する必要があります。

```
/usr/edb/pem/bin/configure-pem-server.sh -t 3 \
                                        -ho 172.16.161.245 \
                                        -p 5444 \
                                        -su pemsuper \
                                        --asb-host-name 127.0.0.1 \
                                        --replication-solution efm \
                                        -srp --efm-cluster-name=efm \
                                        -srp --efm-install-path=/usr/edb/efm-5.2/ \
                                        --replication-role standby \
                                        --db-unitfile edb-as-17 \
                                        --db-install-path /usr/edb/as17/ \
                                        --use-agent-sslcert /path/to/agent/cert \
                                        --use-agent-sslkey /path/to/agent/key \
                                        -aro agent_id=2
```

さまざまな追加の詳細の入力を求められます。構成オプションについては、Configuring the PEM server on Linux を参照してください。

PEM Web アプリケーションの展開

すべての Web アプリケーションホストで次の手順を実行します。

1. PEM パッケージと EPAS クライアントバイナリをインストールします。

```
dnf install edb-pem edb-as17-server-client
```

1. すべてのサーバーのファイアウォールで次のポートを開きます。

- PEM サーバー HTTPS の 8443

例

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

1. PEM Web アプリケーションを構成します。VIP を PEM サーバーアドレスとして指定します。

```
/usr/edb/pem/bin/configure-pem-server.sh -t 2 \
                                        -ho 172.16.161.245 \
                                        -p 5444 \
                                        -su pemsuper \
                                        --db-install-path /usr/edb/as17/
```

共通エージェントユーザを使用する場合、ユーザ名と、そのユーザ用に生成された証明書とキーへのパスを提供する必要があります。

```
/usr/edb/pem/bin/configure-pem-server.sh -t 2 \
                                         -ho 172.16.161.245 \
                                         -p 5444 \
                                         -su pemsuper \
                                         --db-install-path /usr/edb/as17/ \
                                         --use-agent-user common_agent_user \
                                         --use-agent-sslcert /path/to/agent/cert \
                                         --use-agent-sslkey /path/to/agent/key
```

独自の証明書を使用する場合、agent_id を次の整数 N として指定し、証明書へのパスと、対応する agentN ユーザー用に生成されたキーを指定する必要があります。

```
/usr/edb/pem/bin/configure-pem-server.sh -t 2 \
                                         -ho 172.16.161.245 \
                                         -p 5444 \
                                         -su pemsuper \
                                         --db-install-path /usr/edb/as17/ \
                                         --use-agent-sslcert /path/to/agent/cert \
                                         --use-agent-sslkey /path/to/agent/key \
                                         -aro agent_id=4
```

さまざまな追加の詳細の入力を求められます。構成オプションについては、Configuring the PEM server on Linux を参照してください。

1. configure-selinux.sh スクリプトを実行して、PEM の SELinux ポリシーを構成します。

```
USE_NGINX=1 /usr/edb/pem/bin/configure-selinux.sh
```

1. これで、前に作成した pemsuper スーパーユーザーを使用して、Web アプリケーションインスタンスにログインできます。
2. インスタンス間で PEM Web アプリケーションのユーザー設定を同期することを選択した場合、*configuring each web application instance to use the backend for user settings* によって今すぐセットアップを完了します。

Storing user settings in the backend, accessible to all web application instances

PEM Web アプリケーションの複数のインスタンスを展開する場合、ユーザー設定をバックエンドデータベースに保存するように PEM を構成して、すべてのインスタンスで同期されるようにします。

PEM バックエンドの構成

PEM バックエンドの Postgres インスタンスで、スーパーユーザーとして接続し、新しいデータベースを作成します。

新しいユーザーにこのデータベースに接続する権利を付与し、PEM データベースに接続する権利を取り消します。

```
CREATE DATABASE pem_user_settings;
CREATE USER pem_user_settings WITH PASSWORD choose-a-password;
GRANT CONNECT ON DATABASE pem_user_settings TO pem_user_settings;
GRANT CREATE ON DATABASE pem_user_settings TO pem_user_settings;
REVOKE CONNECT ON DATABASE pem FROM pem_user_settings;
```

また、このインスタンス上の他のデータベースたとえば postgres や edb などに接続する権限を取り消す必要があります。

pem_user_settings データベースから、新しいユーザーにスキーマを作成する機能を付与します。

```
GRANT CREATE ON SCHEMA public TO pem_user_settings;
```

また、ユーザー pem_user_settings が Web アプリケーションインスタンスから各 Postgres インスタンスに接続することを許可されていることを確認する必要があります。これには、次の形式のルールを pg_hba.conf に追加する必要がある場合があります。

```
hostssl  pem_user_settings  pem_user_settings  <web-application-subnet>  scram-sha-256
```

スーパーユーザーとして新しいデータベースに接続し、新しいユーザーに新しいデータベースに必要なすべての権限を付与します。

```
ALTER DEFAULT PRIVILEGES FOR ROLE pem_user_settings GRANT SELECT, INSERT, UPDATE,␣
↪DELETE ON TABLES TO pem_user;
ALTER DEFAULT PRIVILEGES FOR ROLE pem_user_settings GRANT SELECT, UPDATE ON SEQUENCES␣
↪TO pem_user;
```

PEM Web アプリケーションの構成

PEM Web アプリケーションの各インスタンスで、 config_local.py ファイルを見つけるか、存在しない場合は作成します Managing configuration settings を参照してください。

次の行を追加し、 <password> を pem_user_settings ユーザーのパスワードに置き換えます。

- 単一のエンドポイントを使用してトラフィックをプライマリ PEM バックエンドインスタンスにルーティングしている場合、<host-ip> および<host-port> はそのエンドポイントの IP アドレスとポートである必要があります。

```
CONFIG_DATABASE_URI="postgresql://pem_user_settings:<password>@/pem_user_settings?
↪host=<host-ip>:<host-port>"
```

- マルチホスト接続文字列を使用してトラフィックを PEM プライマリにルーティングしている場合、次のフォーマットを使用して、各 PEM バックエンドの詳細を指定できます。

```
CONFIG_DATABASE_URI="postgresql://pem_user_settings:<password>@/pem_user_settings?
↪host=<host-1-ip>:<host-1-port>&host=<host-2-ip>:<host-2-port>&target_session_
↪attrs=read-write"
```

HA PEM using the SM architecture with Patroni

このページでは、リファレンスアーキテクチャ SM に従って高可用性 HAPEM 展開をインストールおよび構成するための詳細な手順を提供します。

この例では、クラスター管理に Patroni を使用し、PEM バックエンドデータベースとして PostgreSQL 17 を、プライマリにトラフィックをルーティングするメカニズムとしてマルチホスト文字列を使用します。私たちのオペレーティングシステムは RHEL 9 x86 です。

他のオプションを理解するには、 *High availability patterns for PEM deployment* を参照してください。

また、*Deciding on an SSL model* を参照して、インストールを開始する前に SSL 証明書とキーを提供する方法を決定します。以下の例では、次の IP アドレスを使用します。

- 172.16.161.200-PEM バックエンドプライマリ
- 172.16.161.201-PEM バックエンドスタンバイ 1
- 172.16.161.202-PEM バックエンドスタンバイ 2
- 172.16.161.211-PEM Web アプリケーション 1
- 172.16.161.212-PEM Web アプリケーション 2
- 172.16.161.213-PEM Web アプリケーション 3
- 172.16.161.111 - etcd-node-1
- 172.16.161.112 - etcd-node-2
- 172.16.161.113 - etcd-node-3

etcd を使用した分散コンセンサスストアの設定

Patroni には分散コンセンサスストア DCS が必要です。etcd を使用して DCS を設定している場合、この手順をスキップして、既存の DCS を使用できます。ここでは、HA PEM アプリケーションクラスターとは別に、3 つのホストに最小限の 3 ノード DCS を作成します。

各ノードで

1. PGDG リポジトリから etcd をインストールする

```
dnf install -y https://download.postgresql.org/pub/repos/yum/reporpms/EL-9-x86_64/
↪pgdg-redhat-repo-latest.noarch.rpm
dnf --enablerepo=pgdg-rhel9-extras install -y etcd
```

2. /etc/etcd/etcd.conf の内容を次のものに置き換え、 <node-ip> および node-name の適切な値を置き換えます。例、172.16.161.111 、etcd-node-1 。

```
#[Member]
ETCD_LISTEN_PEER_URLS="http://<node-ip>:2380"
ETCD_LISTEN_CLIENT_URLS="http://localhost:2379,http://<node-ip>:2379"
ETCD_NAME="<node-name>"
#[Clustering]
```

(次のページに続く)

(前のページからの続き)

```
ETCD_INITIAL_ADVERTISE_PEER_URLS="http://<node-ip>:2380"
ETCD_INITIAL_CLUSTER="etcd-node-1=http://172.16.161.111:2380,etcd-node-2=http://172.
↪16.161.112:2380,etcd-node-3=http://172.16.161.113:2380"
ETCD_ADVERTISE_CLIENT_URLS="http://<node-ip>:2379"
ETCD_INITIAL_CLUSTER_TOKEN="etcd-cluster-1"
ETCD_INITIAL_CLUSTER_STATE="new"
```

3. ポート 2380 および 2379 がファイアウォールで開いていることを確認します。

```
firewall-cmd --zone=public --add-port=2380/tcp --permanent
firewall-cmd --zone=public --add-port=2379/tcp --permanent
firewall-cmd --reload
```

4. etcd サービスが再起動され有効になっていることを確認します。

```
systemctl enable etcd
systemctl restart etcd
```

EDB リポジトリの構成

すべてのバックエンドおよび Web アプリケーションノードで、EDB Repos を構成します。詳細については、 Getting started with repositories を参照してください。

```
curl -1sLf  https://downloads.enterprisedb.com/<token>/<repo>/setup.rpm.sh | bash
```

Patroni クラスターを作成する

PEM をインストールする前に、バックエンドホストで動作する Patroni クラスターが必要です。このセクションでは、このようなクラスターを展開する手順を提供します。Patroni クラスターをプロビジョニングする独自のプロセスがある場合、これらの手順に従う必要はありません。ただし、sslutils を使用する予定がある場合は、続行する前にクラスターにインストールされていることを確認する必要があります。

初期パッケージのインストールと Postgres 構成

特に記載がない限り、すべてのバックエンドノードで次の手順を実行します。

1. 次のパッケージをインストールします。

   - *PostgreSQL サーバーへの接続エラー* PEM サーバーのバックエンドデータベース
   - *HA PEM using the SM architecture with Patroni*

```
dnf -qy module disable postgresql
dnf -y install postgresql17-server postgresql17-contrib edb-patroni
```

1. Postgres データベースを初期化し、サービスを開始します。

```
PGSETUP_INITDB_OPTIONS="-E UTF-8" /usr/pgsql-17/bin/postgresql-17-setup initdb
systemctl start postgresql-17
systemctl enable postgresql-17
```

1. ファイアウォールでポート 5432 を開きます。

```
firewall-cmd --zone=public --add-port=5432/tcp --permanent
firewall-cmd --reload
```

オプション sslutils のインストール

PEM が認証局として機能し、クライアント証明書を自動的に発行する場合、*Regenerating the agent SSL certificates for PEM using sslutils* を構成する必要があります。独自の証明書を提供する予定がある場合は、この手順をスキップできます。

1. すべてのバックエンドホストに sslutils パッケージをインストールします。

```
dnf install sslutils_17
```

Patroni ユーザーを作成する

プライマリバックエンドホストのみで次の手順を実行します。

1. Patroni スーパーユーザー、レプリケーション、およびリワインドロールを作成します。スーパーユーザーとして次の SQL を実行します。

```
-- Patroni superuser
CREATE USER patroni_super WITH SUPERUSER ENCRYPTED PASSWORD your-password-here;

-- Patroni replication user
CREATE USER patroni_rep WITH REPLICATION ENCRYPTED PASSWORD your-password-here;

-- Patroni rewind user, if you intend to enable use_pg_rewind in your Patroni␣
↪configuration
CREATE USER patroni_rew WITH ENCRYPTED PASSWORD your-password-here;
GRANT EXECUTE ON function pg_catalog.pg_ls_dir(text, boolean, boolean) TO patroni_rew;
GRANT EXECUTE ON function pg_catalog.pg_stat_file(text, boolean) TO patroni_rew;
GRANT EXECUTE ON function pg_catalog.pg_read_binary_file(text) TO patroni_rew;
GRANT EXECUTE ON function pg_catalog.pg_read_binary_file(text, bigint, bigint,␣
↪boolean) TO patroni_rew;
```

選択したパスワードを入力します。

プライマリで Patroni をセットアップする

1. /etc/patroni/patroni.yml でファイルを作成し、次のコンテンツを追加し、クラスターに一致するように IP アドレスを調整し、必要なクラスターとノード名を指定します。

```
scope: cluster-name
name: node-name

restapi:
    listen: 172.16.161.200:8008
    connect_address: 172.16.161.200:8008

etcd3:
  hosts:
    - 172.16.161.111:2379
    - 172.16.161.112:2379
    - 172.16.161.113:2379

bootstrap:
  dcs:
    ttl: 30
    loop_wait: 10
    retry_timeout: 10
    maximum_lag_on_failover: 1048576
    postgresql:
      use_pg_rewind: true
      pg_hba:
        - # Allow our Patroni users to connect as required
        - host    all             patroni_super   127.0.0.1/32            scram-sha-
↪256
        - host replication        patroni_rep     127.0.0.1/32            scram-sha-
↪256
        - host all                patroni_rew     127.0.0.1/32            scram-sha-
↪256
        - host replication        patroni_rep     172.16.161.200/32       scram-sha-
↪256
        - host all                patroni_rew     172.16.161.200/32       scram-sha-
↪256
        - host replication        patroni_rep     172.16.161.201/32       scram-sha-
↪256
        - host all                patroni_rew     172.16.161.201/32       scram-sha-
↪256
        - host replication        patroni_rep     172.16.161.202/32       scram-sha-
↪256
        - host all                patroni_rew     172.16.161.202/32       scram-sha-
↪256
```

(次のページに続く)

(前のページからの続き)

```
        - # This is the default config for this PG package
        - local   all             all                                     peer
        - host    all             all             127.0.0.1/32            scram-sha-
↪256
        - host    all             all             ::1/128                 scram-sha-
↪256
        - local   replication     all                                     peer
        - host    replication     all             127.0.0.1/32            scram-sha-
↪256
        - host    replication     all             ::1/128                 scram-sha-
↪256

  initdb:
    - encoding: UTF8
    - data-checksums

postgresql:
    listen: *:5432
    connect_address: 172.16.161.200:5432
    data_dir: /var/lib/pgsql/17/data
    bin_dir: /usr/pgsql-17/bin
    authentication:
        replication:
            username: patroni_rep
            password: your-password-here
        superuser:
            username: patroni_super
            password: your-password-here
        rewind:
            username: patroni_rew
            password: your-password-here
    parameters:
        unix_socket_directories: /var/run/postgresql, /tmp
```

1. Postgres サービスを停止して無効にし、Patroni を起動します

```
systemctl disable --now postgresql-17
systemctl enable --now patroni
```

スタンバイノードのセットアップ

各スタンバイノードで

1. Postgres のサービスを停止して無効にします。

```
systemctl disable --now postgresql-17
```

2. データベースサーバーのデータディレクトリを削除します。

```
rm -rf /var/lib/pgsql/17/data
```

3. /etc/patroni/patroni.yml に Patroni 構成ファイルを作成します。プライマリと同じコンテンツを使用しますが、次の変更を行います。
   - bootstrap セクション全体を削除します
   - name を各スタンバイの一意の名前に変更します
   - restapi および postgresql.connect_address の下の IP アドレスを、構成しているスタンバイのアドレスに変更します。
4. Patroni を起動して有効にします。

```
systemctl enable --now patroni
```

これで、機能する Patroni クラスターができましたが、SSL または PEM に必要な特定の HBA ルールはありません。次のセクションでこれらを構成します。

PEM インストール用の Patroni クラスターの準備

HA モードで PEM をインストールする前に、クラスターが 2 つの要件を満たしていることを確認する必要があります。

- PEM エージェントから PEM バックエンドデータベースへの接続に cert 認証方法を使用できるように SSL が構成されます。
- HBA ルールが適用され、エージェントと Web アプリケーションインスタンスが PEM バックエンドに接続できます。

SSL ファイルの準備

Generating SSL files for HA PEM backend hosts の指示に従ってください。

Patroni クラスターを変更して SSL および HBA ルールを追加する

1. バックエンドホストで、patronictl edit-config を使用してクラスターの編集ツールを開きます。たとえば、前のセクションで構築されたクラスターのコマンドは次のとおりです。

```
patronictl -c /etc/patroni/patroni.yml edit-config
```

1. postgresql.parameters の下に次を追加します。

```
ssl: on
ssl_ca_file: root.crt
ssl_cert_file: server.crt
ssl_key_file: server.key
```

CRL がある場合は、次も設定します。

```
ssl_crl_file: root.crl
```

1. 次の HBA ルールを既存のルールの上の postgresql.pg_hba の **top** に追加し、クラスターと一致するように IP アドレスを調整します。

```
- # Allow users of the web interface to connect to pem and postgres databases
- hostssl pem      +pem_user   127.0.0.1/32           scram-sha-256
- hostssl postgres +pem_user   127.0.0.1/32           scram-sha-256
- hostssl pem      +pem_user   172.16.161.211/32      scram-sha-256
- hostssl postgres +pem_user   172.16.161.211/32      scram-sha-256
- hostssl pem      +pem_user   172.16.161.212/32      scram-sha-256
- hostssl postgres +pem_user   172.16.161.212/32      scram-sha-256
- hostssl pem      +pem_user   172.16.161.213/32      scram-sha-256
- hostssl postgres +pem_user   172.16.161.213/32      scram-sha-256

- # Allow the PEM backend server agents to register and connect to send data to PEM␣
↪server
- hostssl pem      +pem_agent  127.0.0.1/32           cert
- hostssl pem      +pem_admin  172.16.161.200/32      scram-sha-256
- hostssl pem      +pem_agent  172.16.161.200/32      cert
- hostssl pem      +pem_admin  172.16.161.201/32      scram-sha-256
- hostssl pem      +pem_agent  172.16.161.201/32      cert
- hostssl pem      +pem_admin  172.16.161.202/32      scram-sha-256
- hostssl pem      +pem_agent  172.16.161.202/32      cert

- # Allow the PEM web app server agents to register and connect to send data to PEM␣
↪server
- hostssl pem      +pem_admin   172.16.161.211/32    scram-sha-256
- hostssl pem      +pem_agent   172.16.161.211/32    scram-sha-256
- hostssl pem      +pem_admin   172.16.161.212/32    scram-sha-256
- hostssl pem      +pem_agent   172.16.161.212/32    scram-sha-256
- hostssl pem      +pem_admin   172.16.161.213/32    scram-sha-256
- hostssl pem      +pem_agent   172.16.161.213/32    scram-sha-256

- # Allow our superuser to connect to perform cluster configuration
- hostssl pem,postgres      pemsuper   172.16.161.202/32    scram-sha-256
- hostssl pem,postgres      pemsuper   172.16.161.201/32    scram-sha-256
- hostssl pem,postgres      pemsuper   172.16.161.213/32    scram-sha-256
- hostssl pem,postgres      pemsuper   172.16.161.211/32    scram-sha-256
- hostssl pem,postgres      pemsuper   172.16.161.200/32    scram-sha-256
- hostssl pem,postgres      pemsuper   172.16.161.212/32    scram-sha-256

- # Allow monitoring agents to connect to PEM
- # Specify a subnet that includes all your monitored servers
```

(次のページに続く)

(前のページからの続き)

```
- # Duplicate these lines for multiple subnets
- hostssl pem      +pem_user   0.0.0.0/0              scram-sha-256
- hostssl pem      +pem_agent  0.0.0.0/0              cert
```

1. 共通エージェントユーザーを使用する場合、PEM のインストール中に最初のエージェントが接続できるように 1 行追加します。この行は、必要に応じて後で削除できます。

```
- hostssl pem      common_agent_user  127.0.0.1/32          cert
```

1. 非 SSL クラスターで開始した場合、既存の HBA ルールを確認し、host ルールを hostssl ルールに変換して、必要に応じてクライアントに安全な接続を使用させることを検討します。

エージェントユーザーと証明書の準備

独自の SSL 証明書を提供する場合は、次の手順に従って PEM 展開の準備をします。**sslutils を使用している場合、このセクションをスキップしてください。**

プライマリで、共通エージェントユーザーを作成します。コモンエージェントユーザーにスーパーユーザー権限を付与します。これは一時的にのみ必要であり、後で取り消されます。

```
CREATE USER common_agent_user WITH SUPERUSER;
```

このユーザーの証明書とキーペアを生成し、すべてのバックエンドおよび Web アプリケーションホストに保存します。

証明書を生成する方法に関するガイダンスについては、*Generating agent certificates manually* を参照してください。

PEM は、3 つのバックエンドと 3 つのフロントエンドホストにわたる展開に対して、一意の番号付けの Postgres ユーザー agent1〜agent6 を自動的に生成します。

この構成をサポートするには、6 つの異なる証明書/キーペア特定のエージェントユーザーごとに 1 つを手動で生成し、各ホストが独自の一意のペアを持つようにそれらを配布する必要があります。

プライマリに PEM をインストールする

プライマリホストでのみ次の手順を実行します。

1. PEM パッケージをインストールします。

```
dnf install edb-pem --allowerasing
```

1. パスワードを使用してログインできるスーパーユーザーを作成します。これは、PEM のインストールに使用するユーザーです。また、PEM バックエンドホストで実行されているエージェントのエージェントサーバーバインディングユーザーにもなります。

```
su - postgres -c "psql -c \"create role pemsuper login superuser password your-
↪password-here;\""
```

1. PEM サーバーを構成するには、3 つのバックエンドホストすべての IP を指定するデータベースオプションオプション 3 を使用して構成スクリプトを実行します。プライマリの外部 IP アドレスではなく、ループバック IP アドレスを提供します。

システムが高可用性 HA 用に正しくセットアップされていることを確認するには、次のフラグを含める必要があります。

- **``--replication-solution patroni`` & ``--replication-role primary``** これらのフラグはスクリプトを強制的に HA モードに互換性し、Patroni.
- **``-srp --patroni-cluster-name`` , ``-srp --patroni-install-path`` & ``-srp --patroni-config-path``** これらのオプションは、PEM が自分のバックエンドクラスターを適切にモニタできることを保証するために使用されます。
- **``--skip-db-certificate``** これにより、HA 環境で証明書を手動で管理しているため、スクリプトが SSL を自動構成しようとするのを防ぎます。

sslutils を使用する場合、`--use-existing-sslutils` を含める必要があります。これにより、スクリプトはデータベース SSL セットアップをスキップしますが、sslutils 拡張機能を利用してエージェント証明書の生成を処理します。

```
/usr/edb/pem/bin/configure-pem-server.sh -t 3 \
                                         -ho 127.0.0.1,172.16.161.201,172.16.161.202 \
                                         -p 5432 \
                                         -su pemsuper \
                                         --asb-host-name 127.0.0.1 \
                                         --replication-solution patroni \
                                         -srp --patroni-cluster-name=<cluster-name> \
                                         -srp --patroni-install-path=/usr/bin/patroni␣
↪\
                                         -srp --patroni-config-path=/etc/patroni/
↪patroni.yml \
                                         --replication-role primary \
                                         --skip-db-certificate \
                                         --use-existing-sslutils \
                                         --db-install-path /usr/pgsql-17/ \
                                         --db-unitfile patroni
```

共通エージェントユーザを使用する場合、ユーザ名と、そのユーザ用に生成された証明書とキーへのパスを提供する必要があります。

```
/usr/edb/pem/bin/configure-pem-server.sh -t 3 \
                                         -ho 127.0.0.1,172.16.161.201,172.16.161.202 \
                                         -p 5432 \
                                         -su pemsuper \
                                         --asb-host-name 127.0.0.1 \
                                         --replication-solution patroni \
                                         -srp --patroni-cluster-name=<cluster-name> \
                                         -srp --patroni-install-path=/usr/bin/patroni␣
```

(次のページに続く)

(前のページからの続き)

```
↪\
                                        -srp --patroni-config-path=/etc/patroni/
↪patroni.yml \
                                        --replication-role primary \
                                        --db-unitfile patroni \
                                        --db-install-path /usr/pgsql-17/ \
                                        --skip-db-certificate \
                                        --use-agent-user common_agent_user \
                                        --use-agent-sslcert /path/to/agent/cert \
                                        --use-agent-sslkey /path/to/agent/key
```

このコマンドを実行した後、共通エージェントユーザーに pem_agent ロールを付与し、スーパーユーザーステータスを取り消します。

```
GRANT pem_agent to common_agent_user;
ALTER USER common_agent_user WITH NOSUPERUSER;
```

+pem_agent ルールと一致するため、必要に応じて、この段階で共通エージェントユーザーに追加した HBA ルールを削除することもできます。

独自の証明書を使用する場合、 agent_id を 1 として指定し、対応する agent1 ユーザー用に生成された証明書とキーへのパスを指定する必要があります。

```
/usr/edb/pem/bin/configure-pem-server.sh -t 3 \
                                        -ho 127.0.0.1,172.16.161.201,172.16.161.202 \
                                        -p 5432 \
                                        -su pemsuper \
                                        --asb-host-name 127.0.0.1 \
                                        --replication-solution patroni \
                                        -srp --patroni-cluster-name=<cluster-name> \
                                        -srp --patroni-install-path=/usr/bin/patroni␣
↪\
                                        -srp --patroni-config-path=/etc/patroni/
↪patroni.yml \
                                        --replication-role primary \
                                        --db-unitfile patroni \
                                        --db-install-path /usr/pgsql-17/ \
                                        --skip-db-certificate \
                                        --use-agent-sslcert /path/to/agent/cert \
                                        --use-agent-sslkey /path/to/agent/key \
                                        -aro agent_id=1
```

さまざまな追加の詳細の入力を求められます。構成オプションについては、Configuring the PEM server on Linux を参照してください。

スタンバイに PEM をインストールする

スタンバイホストのみで次の手順を実行します。

1. PEM パッケージをインストールします。

```
dnf install edb-pem --allowerasing
```

1. PEM サーバーを構成するには、データベースオプションオプション 3 を使用して構成スクリプトを実行し、ホストを仮想 IPVIP に設定します。

高可用性 HA モードを正しく有効にするには、次の特定のフラグを含める必要があります。

- ``**--replication-solution patroni**`` 構成を Patroni と統合します。
- ``**--replication-role standby**`` HA クラスター内でのこのノードの初期ロールを指定します

```
/usr/edb/pem/bin/configure-pem-server.sh -t 3 \
                                         -ho 172.16.161.200,127.0.0.1,172.16.161.202 \
                                         -p 5432 \
                                         -su pemsuper \
                                         --asb-host-name 127.0.0.1 \
                                         --replication-solution patroni \
                                         -srp --patroni-cluster-name=<cluster-name> \
                                         -srp --patroni-install-path=/usr/bin/patroni␣
↪\
                                         -srp --patroni-config-path=/etc/patroni/
↪patroni.yml \
                                         --replication-role standby \
                                         --db-unitfile patroni \
                                         --db-install-path /usr/pgsql-17/
```

共通エージェントユーザを使用する場合、ユーザ名と、そのユーザ用に生成された証明書とキーへのパスを提供する必要があります。

```
/usr/edb/pem/bin/configure-pem-server.sh -t 3 \
                                         -ho 172.16.161.200,127.0.0.1,172.16.161.202 \
                                         -p 5432 \
                                         -su pemsuper \
                                         --asb-host-name 127.0.0.1 \
                                         --replication-solution patroni \
                                         -srp --patroni-cluster-name=<cluster-name> \
                                         -srp --patroni-install-path=/usr/bin/patroni␣
↪\
                                         -srp --patroni-config-path=/etc/patroni/
↪patroni.yml \
                                         --replication-role standby \
                                         --db-unitfile patroni \
                                         --db-install-path /usr/pgsql-17/ \
```

(次のページに続く)

(前のページからの続き)

```
                                        --use-agent-user common_agent_user \
                                        --use-agent-sslcert /path/to/agent/cert \
                                        --use-agent-sslkey /path/to/agent/key
```

独自の証明書を使用する場合、agent_id を次の整数 N として指定し、証明書へのパスと、対応する agentN ユーザー用に生成されたキーを指定する必要があります。

```
/usr/edb/pem/bin/configure-pem-server.sh -t 3 \
                                        -ho 172.16.161.200,127.0.0.1,172.16.161.202 \
                                        -p 5432 \
                                        -su pemsuper \
                                        --asb-host-name 127.0.0.1 \
                                        --replication-solution patroni \
                                        -srp --patroni-cluster-name=<cluster-name> \
                                        -srp --patroni-install-path=/usr/bin/patroni␣
↪\
                                        -srp --patroni-config-path=/etc/patroni/
↪patroni.yml \
                                        --replication-role standby \
                                        --db-unitfile patroni \
                                        --db-install-path /usr/pgsql-17/ \
                                        --use-agent-sslcert /path/to/agent/cert \
                                        --use-agent-sslkey /path/to/agent/key \
                                        -aro agent_id=2
```

さまざまな追加の詳細の入力を求められます。構成オプションについては、Configuring the PEM server on Linux を参照してください。

PEM Web アプリケーションの展開

すべての Web アプリケーションホストで次の手順を実行します。

1. PEM パッケージと Postgres クライアントバイナリ用をインストールします。

```
dnf install edb-pem postgresql17-server
```

1. すべてのサーバーのファイアウォールで次のポートを開きます。

- PEM サーバー HTTPS の 8443

例

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

1. PEM Web アプリケーションを構成します。プライマリとすべてのスタンバイを含むマルチホスト文字列を PEM サーバーアドレスとして提供します。

```
/usr/edb/pem/bin/configure-pem-server.sh -t 2 \
                                        -ho 172.16.161.200,172.16.161.201,172.16.161.
↪202 \
                                        -p 5432 \
                                        -su pemsuper \
                                        --db-install-path /usr/pgsql-17/
```

共通エージェントユーザを使用する場合、ユーザ名と、そのユーザ用に生成された証明書とキーへのパスを提供する必要があります。

```
/usr/edb/pem/bin/configure-pem-server.sh -t 2 \
                                        -ho 172.16.161.200,172.16.161.201,172.16.161.
↪202 \
                                        -p 5432 \
                                        -su pemsuper \
                                        --db-install-path /usr/pgsql-17/ \
                                        --use-agent-user common_agent_user \
                                        --use-agent-sslcert /path/to/agent/cert \
                                        --use-agent-sslkey /path/to/agent/key
```

独自の証明書を使用する場合、agent_id を次の整数 N として指定し、証明書へのパスと、対応する agentN ユーザー用に生成されたキーを指定する必要があります。

```
/usr/edb/pem/bin/configure-pem-server.sh -t 2 \
                                        -ho 172.16.161.200,172.16.161.201,172.16.161.
↪202 \
                                        -p 5432 \
                                        -su pemsuper \
                                        --db-install-path /usr/pgsql-17/ \
                                        --use-agent-sslcert /path/to/agent/cert \
                                        --use-agent-sslkey /path/to/agent/key \
                                        -aro agent_id=4
```

さまざまな追加の詳細の入力を求められます。 構成オプションについては、Configuring the PEM server on Linux を参照してください。

1. configure-selinux.sh スクリプトを実行して、PEM の SELinux ポリシーを構成します。

```
USE_NGINX=1 /usr/edb/pem/bin/configure-selinux.sh
```

1. これで、前に作成した pemsuper スーパーユーザーを使用して、Web アプリケーションインスタンスにログインできます。
2. インスタンス間で PEM Web アプリケーションのユーザー設定を同期することを選択した場合、*configuring each web application instance to use the backend for user settings* によって今すぐセットアップを完了します。

Generating SSL files for HA PEM backend hosts

このページでは、 HA PEM バックエンドの SSL ファイルを生成する方法について説明します。SSL モデルの選択については、 *Deciding on an SSL model* を参照してください。

外部 CA を使用する場合

!!!注

SSL 対応の Postgres クラスターをプロビジョニングする既存のプロセスがある場合、これらの条件は既に満たされている可能性があります。この場合、それ以上のアクションは必要ありません。

後続のインストール手順では、これらのファイルがそれぞれ `server.crt` 、`server.key` 、および `root.crt` と呼ばれることを前提としています。これらの名前を使用する必要はありませんが、Postgres 構成が正しいファイル名を参照していることを確認する必要があります。

ここで使用される CA は、PEM エージェントのクライアント証明書の署名に使用したものと同じであることが重要です。それ以外の場合、`cert` 認証は失敗します。

オプションで、証明書失効リスト CRL が利用可能な場合は、それを含めることができます。

これらのファイルが `postgres` または `enterprisedb` ユーザーとグループが所有していること、および権限が `0600` に設定されていることを確認します。

sslutils を使用する場合

HA 展開では、PEM は、クラスターマネージャーとの衝突を回避するためにクラスターを構成または再起動しません。したがって、sslutils は手動で構成する必要があります。

!!!注 これらの手順は `postgres` または `enterprisedb` ユーザーとして実行して、ファイルの所有権が正しいことを確認します。

1. 認証局として機能するときに sslutils が使用する単一の証明書とキーペアを生成します。これらを各バックエンドホストの PGDATA に配置します。

```
openssl genrsa -out ca_key.key 4096

openssl req -x509 -new -nodes -key ca_key.key -sha256 -days 3650 \
-out ca_certificate.crt \
-subj "/C=.../ST=.../L=.../O=.../CN=PEM CA"
```

1. 各バックエンドホストで、`ca_certificate.crt` のコピーを作成し、`root.crt` という名前を付けます。これも PGDATA に保存する必要があります。

2. 各バックエンドホストで、サーバーキーを生成し、CA によって署名されたサーバー証明書を生成します。クライアント接続に `verify-full` SSL モードを使用する予定がある場合、共通名がサーバーの DNS 名と一致することが重要です。

```
openssl genrsa -out server.key 4096

openssl req -new -key server.key -out server.csr \
```

(次のページに続く)

(前のページからの続き)

```
    -subj "/C=.../ST=.../L=.../O=.../CN=..."

openssl x509 -req -in server.csr -CA ca_certificate.crt -CAkey ca_key.key \
    -CAcreateserial -out server.crt -days 3650 -sha256
```

1. キーファイルと証明書ファイルのアクセス許可を設定し、CSR を削除します。

```
chmod 0600 ca_certificate.crt
chmod 0600 ca_key.key
chmod 0600 root.crt
chmod 0600 server.crt
chmod 0600 server.key
rm server.csr
```

ローカル CA を使用する場合

sslutils を使用したくないが、組織の CA がない場合は、PEM ホストまたは他の場所でローカル CA を手動で構成し、それを使用して PEM サーバーとエージェントの証明書を発行できます。

注釈

Postgres クラスターがプロビジョニング中に SSL を使用するように既に構成されている場合、次の手順に従って、既存の SSL 証明書とキーをローカル CA によって署名されたものに置き換えて、Postgres がクライアント証明書を検証できることを確認する必要があります。その逆も可能です。

1. CA をホストするマシンで、単一の証明書とキーペアを生成します。これが CA 証明書とキーになります。

```
openssl genrsa -out ca_key.key 4096

openssl req -x509 -new -nodes -key ca_key.key -sha256 -days 3650 \
-out ca_certificate.crt \
-subj "/C=.../ST=.../L=.../O=.../CN=PEM CA"
```

重要

ローカル CA キーと証明書を安全に保ちます。PEM エージェントのクライアント証明書を生成するために必要です。 !!!

1. 各 PEM バックエンドホストで、サーバーキーを生成し、証明書署名要求 CSR を生成します。クライアント接続に `verify-full` SSL モードを使用する予定がある場合、共通名がサーバーの DNS 名と一致することが重要です。

```
openssl genrsa -out server.key 4096
```

(次のページに続く)

(前のページからの続き)

```
openssl req -new -key server.key -out server.csr \
    -subj "/C=.../ST=.../L=.../O=.../CN=..."
```

1. 各 CSR をローカル CA ホストにコピーし、それを使用してサーバー証明書を生成します。

```
openssl x509 -req -in server.csr -CA ca_certificate.crt -CAkey ca_key.key \
    -CAcreateserial -out server.crt -days 3650 -sha256
```

必要に応じて、PEM ホストから CSR を削除できます。

1. 生成された各サーバー証明書を、その CSR が発生した PEM ホストにコピーします。サーバー証明書とサーバーキーの両方を PGDATA に配置します。
2. ローカル CA パブリック証明書 `ca_certificate.crt` を各 PEM ホストにコピーして PGDATA に配置し、 `root.crt` に名前を変更します
3. 3 つのファイルすべてが Postgres OS ユーザー `postgres` または `enterprisedb` と同じ名前のグループが所有していることを確認します。

```
chown <user>:<group> root.crt
chown <user>:<group> server.crt
chown <user>:<group> server.key
rm server.csr
```

1. キーファイルと証明書ファイルのアクセス許可を設定します。

```
chmod 0600 root.crt
chmod 0600 server.crt
chmod 0600 server.key
```

6.2 Installing Postgres Enterprise Manager agent

リンクを選択して、該当するインストール手順にアクセスします。

6.2.1 Linux x86-64 (amd64)

6.2.1.1 Red Hat Enterprise Linux RHEL とその派生

- *Installing Postgres Enterprise Manager agent on RHEL 10 or OL 10 arm64* 、*Installing Postgres Enterprise Manager agent on RHEL 9 or OL 9 arm64* 、 *Installing Postgres Enterprise Manager agent on RHEL 8 ppc64le*

- *Installing Postgres Enterprise Manager agent on RHEL 10 or OL 10 arm64* 、*Installing Postgres Enterprise Manager agent on RHEL 9 or OL 9 arm64* 、 *Installing Postgres Enterprise Manager agent on RHEL 8 ppc64le*

- *Installing Postgres Enterprise Manager agent on AlmaLinux 9 or Rocky* 、 *Installing Postgres Enterprise Manager agent on AlmaLinux 8 or Rocky*

- *Installing Postgres Enterprise Manager agent on AlmaLinux 9 or Rocky* 、 *Installing Postgres Enterprise Manager agent on AlmaLinux 8 or Rocky*

6.2.1.2 SUSE Linux Enterprise SLES

- *Installing Postgres Enterprise Manager agent on SLES 15 ppc64le*

6.2.1.3 Debian とデリバティブ

- *Installing Postgres Enterprise Manager agent on Ubuntu 24.04 x86_64* 、 *Installing Postgres Enterprise Manager agent on Ubuntu 22.04 x86_64*
- *Installing Postgres Enterprise Manager agent on Debian 12 arm64*、*Installing Postgres Enterprise Manager agent on Debian 11 x86_64*

6.2.2 Linux Installing Postgres Enterprise Manager server on Linux IBM Power (ppc64le)

6.2.2.1 Red Hat Enterprise Linux RHEL とその派生

- *Installing Postgres Enterprise Manager agent on RHEL 9 or OL 9 arm64* 、 *Installing Postgres Enterprise Manager agent on RHEL 8 ppc64le*

6.2.2.2 SUSE Linux Enterprise SLES

- *Installing Postgres Enterprise Manager agent on SLES 15 ppc64le*

6.2.3 Linux Installing Postgres Enterprise Manager server on Linux AArch64 (ARM64)

6.2.3.1 Red Hat Enterprise Linux RHEL とその派生

- *Installing Postgres Enterprise Manager agent on RHEL 9 or OL 9 arm64*
- *Installing Postgres Enterprise Manager agent on RHEL 9 or OL 9 arm64*

6.2.3.2 Debian とデリバティブ

- *Installing Postgres Enterprise Manager agent on Debian 12 arm64*

6.2.4 Windows

- *Installing Postgres Enterprise Manager agent on Windows*

6.2.4.1 Installing Postgres Enterprise Manager agent on Linux x86 (amd64)

オペレーティングシステム固有のインストール手順は、対応するドキュメントに記載されています。

Red Hat Enterprise Linux RHEL とその派生

- *Installing Postgres Enterprise Manager agent on RHEL 10 or OL 10 arm64*
- *Installing Postgres Enterprise Manager agent on RHEL 9 or OL 9 arm64*
- *Installing Postgres Enterprise Manager agent on RHEL 8 ppc64le*
- *Installing Postgres Enterprise Manager agent on RHEL 10 or OL 10 arm64*
- *Installing Postgres Enterprise Manager agent on RHEL 9 or OL 9 arm64*
- *Installing Postgres Enterprise Manager agent on RHEL 8 ppc64le*
- *Installing Postgres Enterprise Manager agent on AlmaLinux 9 or Rocky*
- *Installing Postgres Enterprise Manager agent on AlmaLinux 8 or Rocky*
- *Installing Postgres Enterprise Manager agent on AlmaLinux 9 or Rocky*
- *Installing Postgres Enterprise Manager agent on AlmaLinux 8 or Rocky*

SUSE Linux Enterprise SLES

- *Installing Postgres Enterprise Manager agent on SLES 15 ppc64le*

Debian とデリバティブ

- *Installing Postgres Enterprise Manager agent on Ubuntu 24.04 x86_64*
- *Installing Postgres Enterprise Manager agent on Ubuntu 22.04 x86_64*
- *Installing Postgres Enterprise Manager agent on Debian 12 arm64*
- *Installing Postgres Enterprise Manager agent on Debian 11 x86_64*

Installing Postgres Enterprise Manager agent on RHEL 10 or OL 10 x86_64

前提条件

インストールプロセスを開始する前に

- EDB リポジトリの設定をします。詳細については、getting started with repositories を参照してください。

パッケージをインストールする

```
sudo dnf -y install edb-pem-agent
```

PEM エージェントをインストールした後、PEM エージェントを登録する必要があります。詳細については、*Registering a PEM agent* を参照してください。

Installing Postgres Enterprise Manager agent on RHEL 9 or OL 9 x86_64

前提条件

インストールプロセスを開始する前に

- EDB リポジトリの設定をします。詳細については、getting started with repositories を参照してください。

パッケージをインストールする

```
sudo dnf -y install edb-pem-agent
```

PEM エージェントをインストールした後、PEM エージェントを登録する必要があります。詳細については、*Registering a PEM agent* を参照してください。

Installing Postgres Enterprise Manager agent on RHEL 8 or OL 8 x86_64

前提条件

インストールプロセスを開始する前に

- EDB リポジトリの設定をします。詳細については、getting started with repositories を参照してください。

パッケージをインストールする

```
sudo dnf -y install edb-pem-agent
```

PEM エージェントをインストールした後、PEM エージェントを登録する必要があります。詳細については、*Registering a PEM agent* を参照してください。

Installing Postgres Enterprise Manager agent on AlmaLinux 9 or Rocky

前提条件

インストールプロセスを開始する前に

- EDB リポジトリの設定をします。詳細については、getting started with repositories を参照してください。

パッケージをインストールする

```
sudo dnf -y install edb-pem-agent
```

PEM エージェントをインストールした後、PEM エージェントを登録する必要があります。詳細については、*Registering a PEM agent* を参照してください。

Installing Postgres Enterprise Manager agent on AlmaLinux 8 or Rocky

前提条件

インストールプロセスを開始する前に

- EDB リポジトリの設定をします。詳細については、getting started with repositories を参照してください。

パッケージをインストールする

```
sudo dnf -y install edb-pem-agent
```

PEM エージェントをインストールした後、PEM エージェントを登録する必要があります。詳細については、*Registering a PEM agent* を参照してください。

Installing Postgres Enterprise Manager agent on SLES 15 x86_64

> **注釈**
>
> Postgres Enterprise Manager 8.3 以降は SLES でサポートされています。

前提条件

インストールプロセスを開始する前に

- EDB リポジトリの設定をします。詳細については、getting started with repositories を参照してください。
- 必要な SUSE モジュールをアクティブにします。

```
# You can use SLES 15 SP3 for PEM 8.3 and later:
sudo SUSEConnect -p PackageHub/15.3/x86_64

# You can use SLES 15 SP4 for PEM 8.6 and later:
sudo SUSEConnect -p PackageHub/15.4/x86_64
```

- メタデータを更新します。

```
sudo zypper refresh
```

パッケージをインストールする

```
sudo zypper -n install edb-pem-agent
```

PEM エージェントをインストールした後、PEM エージェントを登録する必要があります。詳細については、*Registering a PEM agent* を参照してください。

Installing Postgres Enterprise Manager agent on Ubuntu 24.04 x86_64

前提条件

インストールプロセスを開始する前に

- EDB リポジトリの設定をします。詳細については、getting started with repositories を参照してください。

パッケージをインストールする

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

PEM エージェントをインストールした後、PEM エージェントを登録する必要があります。詳細については、*Registering a PEM agent* を参照してください。

Installing Postgres Enterprise Manager agent on Ubuntu 22.04 x86_64

前提条件

インストールプロセスを開始する前に

- EDB リポジトリの設定をします。詳細については、getting started with repositories を参照してください。

パッケージをインストールする

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

PEM エージェントをインストールした後、PEM エージェントを登録する必要があります。詳細については、*Registering a PEM agent* を参照してください。

Installing Postgres Enterprise Manager agent on Debian 12 x86_64

前提条件

インストールプロセスを開始する前に

- EDB リポジトリの設定をします。詳細については、getting started with repositories を参照してください。

パッケージをインストールする

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

PEM エージェントをインストールした後、PEM エージェントを登録する必要があります。詳細については、*Registering a PEM agent* を参照してください。

Installing Postgres Enterprise Manager agent on Debian 11 x86_64

前提条件

インストールプロセスを開始する前に

- EDB リポジトリの設定をします。詳細については、getting started with repositories を参照してください。

パッケージをインストールする

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

PEM エージェントをインストールした後、PEM エージェントを登録する必要があります。詳細については、*Registering a PEM agent* を参照してください。

6.2.4.2 Installing Postgres Enterprise Manager agent on Linux AArch64 (ARM64)

オペレーティングシステム固有のインストール手順は、対応するドキュメントに記載されています。

Red Hat Enterprise Linux RHEL とその派生

- *Installing Postgres Enterprise Manager agent on RHEL 9 or OL 9 arm64*
- *Installing Postgres Enterprise Manager agent on RHEL 9 or OL 9 arm64*

Debian とデリバティブ

- *Installing Postgres Enterprise Manager agent on Debian 12 arm64*

Installing Postgres Enterprise Manager agent on RHEL 9 or OL 9 arm64

前提条件

インストールプロセスを開始する前に

- EDB リポジトリの設定をします。詳細については、getting started with repositories を参照してください。

パッケージをインストールする

```
sudo dnf -y install edb-pem-agent
```

PEM エージェントをインストールした後、PEM エージェントを登録する必要があります。詳細については、*Registering a PEM agent* を参照してください。

Installing Postgres Enterprise Manager agent on Debian 12 arm64

前提条件

インストールプロセスを開始する前に

- EDB リポジトリの設定をします。詳細については、getting started with repositories を参照してください。

パッケージをインストールする

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

PEM エージェントをインストールした後、PEM エージェントを登録する必要があります。詳細については、*Registering a PEM agent* を参照してください。

Installing Postgres Enterprise Manager agent on RHEL 10 or OL 10 arm64

前提条件

インストールプロセスを開始する前に

- EDB リポジトリの設定をします。詳細については、getting started with repositories を参照してください。

パッケージをインストールする

```
sudo dnf -y install edb-pem-agent
```

PEM エージェントをインストールした後、PEM エージェントを登録する必要があります。詳細については、*Registering a PEM agent* を参照してください。

6.2.4.3 Installing Postgres Enterprise Manager agent on Linux IBM Power (ppc64le)

オペレーティングシステム固有のインストール手順は、対応するドキュメントに記載されています。

Red Hat Enterprise LinuxRHEL

- *Installing Postgres Enterprise Manager agent on RHEL 9 or OL 9 arm64*
- *Installing Postgres Enterprise Manager agent on RHEL 8 ppc64le*

SUSE Linux Enterprise SLES

- *Installing Postgres Enterprise Manager agent on SLES 15 ppc64le*

Installing Postgres Enterprise Manager agent on RHEL 9 ppc64le

前提条件

インストールプロセスを開始する前に

- EDB リポジトリの設定をします。詳細については、getting started with repositories を参照してください。

パッケージをインストールする

```
sudo dnf -y install edb-pem-agent
```

PEM エージェントをインストールした後、PEM エージェントを登録する必要があります。詳細については、*Registering a PEM agent* を参照してください。

Installing Postgres Enterprise Manager agent on RHEL 8 ppc64le

前提条件

インストールプロセスを開始する前に

- EDB リポジトリの設定をします。詳細については、getting started with repositories を参照してください。

パッケージをインストールする

```
sudo dnf -y install edb-pem-agent
```

PEM エージェントをインストールした後、PEM エージェントを登録する必要があります。詳細については、*Registering a PEM agent* を参照してください。

Installing Postgres Enterprise Manager agent on SLES 15 ppc64le

> **注釈**
>
> Postgres Enterprise Manager 8.3 以降は SLES でサポートされています。

前提条件

インストールプロセスを開始する前に

- EDB リポジトリの設定をします。詳細については、getting started with repositories を参照してください。
- 必要な SUSE モジュールをアクティブにします。

```
# You can use SLES 15 SP3 for PEM 8.3 and later:
sudo SUSEConnect -p PackageHub/15.3/ppc64le
```

(次のページに続く)

(前のページからの続き)

```
# You can use SLES 15 SP4 for PEM 8.6 and later:
sudo SUSEConnect -p PackageHub/15.4/ppc64le
```

- メタデータを更新します。

```
sudo zypper refresh
```

パッケージをインストールする

```
sudo zypper -n install edb-pem-agent
```

PEM エージェントをインストールした後、PEM エージェントを登録する必要があります。詳細については、*Registering a PEM agent* を参照してください。

Installing Postgres Enterprise Manager agent on RHEL 10 ppc64le

前提条件

インストールプロセスを開始する前に

- EDB リポジトリの設定をします。詳細については、getting started with repositories を参照してください。

パッケージをインストールする

```
sudo dnf -y install edb-pem-agent
```

PEM エージェントをインストールした後、PEM エージェントを登録する必要があります。詳細については、*Registering a PEM agent* を参照してください。

6.2.4.4 Installing Postgres Enterprise Manager agent on Windows

Windows の PEM エージェントグラフィカルインストーラーは、PEM エージェントをインストールおよび登録します。

PEM エージェントインストーラーを起動するには、管理者権限を引き受け、インストーラーが含まれるディレクトリに移動します。次に、インストーラーを呼び出します。

```
pem_agent-9.<x>.<x>-<x>-platform.exe
```

1. [Setup] 画面が開き、PEM エージェントインストーラーが開きます。**Next** を選択します。
2. 適切なラジオボタンを選択して契約に同意する前に、ライセンス契約を確認してください。**Next** を選択します。
3. デフォルトでは、PEM エージェントは `C:\Program Files (x86)\edb\pem` ディレクトリにインストールされます。デフォルトのインストールディレクトリをそのまま使用できます。または、**インス**

トールディレクトリフィールドの内容を変更し、PEM エージェントの代替インストールディレクトリを指定できます。

- デフォルトでは、PEM エージェントインストーラーは、管理者の`%APPDATA%\pem` ディレクトリに証明書を配置します。インストールプロセスに証明書ファイルの代替パスを指定できるダイアログボックスを含めるには、**Show advanced options** を選択します。
- **Register now?** を選択して、新しくインストールされた PEM エージェントを PEM サーバーに登録します.
- **Next** を選択します。

4. [PEM Server Installation Details] ダイアログボックスに PEM サーバーの接続の詳細を入力します。
   - **Host** フィールドに PEM データベースサーバーが存在するシステムの名前または IP アドレスを指定します。PEM-HTTPD Web サーバーと PEM データベースが別のシステムでホストされている場合、PEM データベースのホストを指定する必要があります。
   - **User Name** フィールドにデータベーススーパーユーザーの名前を指定します。
   - に関連付けられたパスワードを指定します **Password** フィールドにデータベースのスーパーユーザー。
   - PostgreSQL が **Port** フィールドで監視しているポートを指定します。
5. **Next** を選択して、pemAgent サービスアカウントに進みます。インストーラーはサーバーに接続して、詳細が正しいことを確認しようとします。

> **注釈**
>
> PEM サーバーは、PEM エージェントインストーラーからの接続を許可する必要があります。接続エラーが発生した場合は、[PEM Server Installation Details] ダイアログボックスで指定した接続プロパティが正しいことを確認します。 PEM サーバー上の `pg_hba.conf` ファイルで、エラーメッセージに記載されているサーバーへの接続が許可されていることを確認します。

1. 実行する pemAgent サービスの edb アカウントのパスワードを入力します。エージェント証明書とキーファイルは `C:\Users\edb\AppData\Roaming\pem` ディレクトリに作成されます。 **Next** を選択します。
2. PEM Web インターフェースの [ブラウザ] パネルに表示されるツリーには、**Description** フィールドに入力された値が表示され、PEM エージェントを識別します。エージェントがインストールされているマシンのホスト名、またはホストの機能を反映した名前など、エージェントの説明的な名前を指定します。

わかりやすい名前を指定するか、PEM エージェントホストが提供するデフォルトを受け入れます。**Next** を選択します。

1. **Show advanced options** を選択した場合、 Advanced Options ダイアログボックスが開きます。デフォルトでは、PEM エージェントインストーラーは証明書を `C:\Program Files (x86)\edb\pem` ディレクトリに配置します。証明書の代替パスを指定するか、デフォルトを受け入れて **Next** を選択します。
2. ウィザードで PEM エージェントをインストールする準備が整います。**Next** を選択します。

3. [Ready to Install] ダイアログボックスで、[**Next**] を選択します。インストーラーはファイルをシステムにコピーし、PEM サーバーにエージェントを登録します。

インストールが完了すると、PEM エージェントが実行され、オペレーティングシステムとホストデータが PEM サーバーに報告されます。PEM エージェントのホストで Postgres インスタンスの監視を開始するには、PEM のエンタープライズディレクトリに追加し、エージェントにバインドします。

コマンドインからエージェントインストーラーの呼び出し

PEM エージェントインストーラーのコマンドラインオプションは、リソースまたはシステム構成が制限されているためにグラフィカルインストールが機能しない場合に機能を提供します。次のことができます。

- インストーラーを呼び出すときに`--mode unattended` オプションを含めて、ユーザー入力なしでインストールを実行します。

すべてのコマンドラインオプションがすべてのプラットフォームに適しているわけではありません。コマンドラインオプションの完全なリファレンスガイドについては、インストーラーを起動するときに`--help` オプションを含めます。

無人モードでの PEM エージェントインストーラーの呼び出し

インストーラーを呼び出すときにコマンドラインでインストール設定を指定することにより、PEM サーバーの無人インストールを実行できます。PEM サーバーをインストールするシステムはインターネットにアクセスできること必要です。

無人モードで PEM エージェントインストーラーを呼び出す前に、以下を行う必要があります。

- PEM サーバーをインストールします。 PEM サーバーの `pg_hba.conf` ファイルは、PEM エージェントのホストからの接続を許可する必要があります。
- 監視対象の Postgres データベースで SSL が有効になっていて、接続を受け入れていることを確認します。

PEM エージェントをインストールするには管理者権限が必要です。次のコマンドを使用して、PEM エージェントインストーラーを無人モードで呼び出します。

```
pem-agent-9.<x>.<x>-windows-x64.exe --mode unattended
- -pghost <pem_server_host_address> --pgport <pem_server_port>
- -pguser postgres --pgpassword <pguser_password>
- -agent_description <agent_name>
```

そこで

- `pem_server_host_address` は、PEM サーバーのホストの IP アドレスを指定します。
- `pem_server_port` は、バッキング PEM データベースが使用するポートを指定します。デフォルトでは、データベースはポート 5432 を使用します。
- `pguser_password` は、PEM データベースのスーパーユーザーに関連付けられたパスワードを指定します。
- `agent_name` は、PEM エージェントの説明的な名前を指定します。

> **注釈**
>
> Windows 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、*agent.cfg* ファイルで `AllowBatchJobSteps` パラメーターを `True` に設定します。 PEM エージェントは、デフォルトではバッチ/シェルスクリプトを実行しません。

6.3 Configuring user authentication in PEM

PEM ユーザー認証は Postgres バックエンドサーバーに委任されます。特定のユーザー名で PEM にログインできるようにするには、バックエンドにその名前の Postgres ユーザーが存在する必要があります。Web アプリケーションホストから `pem` データベースに接続するには、ユーザーが `pem_user` ロールを持ち、`pg_hba.conf` によって許可されている必要があります。PEM ユーザーロールの詳細については、*PEM 認証の管理* を参照してください。

デフォルトでは、PEM はユーザーにパスワードを使用した認証を必要とします。オプションで 2FA を適用したり、Kerberos を使用したパスワードレス認証を有効にしたりできます。

- *1. 二要素認証 2FA*
- *Kerberos authentication on Linux*
- *Kerberos authentication on Windows*

6.3.1 Configuring the PEM server to use two-factor authentication

PEM は、二要素認証 2FA の 2 つの方法をサポートしています。

- メール認証
- 認証アプリ Google Authenticator など

2FA を有効にするには、これらの設定を `config.py` ファイルから `config_local.py` ファイルにコピーし、次のパラメーターを変更できます。

Parameter	Description
MFA_ENABLED	二要素認証を有効にするには、`true`に設定します。デフォルト値は`false`です。
MFA_FORCE_REGISTRA	ユーザーにログイン時に二要素認証方法を強制的に登録するように求めるには`true`に設定します。デフォルト値は`false`です。
MFA_SUPPORTED_METI	メール認証方法を使用する場合は`email`に設定し、TOTP ベースのアプリケーション認証方法を使用する場合は`authenticator`に設定します。
MFA_EMAIL_SUBJECT	メール認証のメールの件名に設定します。デフォルト値は`<APP_NAME> - Verification Code`です。

6.3.1.1 メールサーバーの構成

メール認証方法を使用するには、メールサーバーの設定を構成する必要があります。

PEM サーバーは、PEM 構成に保存された SMTP 構成、または Flask-Mail を使用して電子メールを送信できます。

PEM 構成から内部 SMTP 構成を使用してメール確認コードを送信するには、パラメーター `MAIL_USE_PEM_INTERNAL` `True` に設定します。 `False` に設定している場合、次のメール構成を使用してコードをユーザー指定の電子メールアドレスに送信します。

- MAIL_SERVER = 'localhost'
- MAIL_PORT = 25
- MAIL_USE_TLS = False
- MAIL_USE_SSL = False
- MAIL_USERNAME = なし
- MAIL_PASSWORD = なし
- MAIL_DEFAULT_SENDER = なし

これらの構成の詳細については、 Flask-Mail documentation を参照してください。

注釈

PEM SMTP アラートはこの構成を使用しません。

6.3.2 Configuring the PEM server to use Kerberos authentication

PEM サーバーの Kerberos 認証を構成できます。Kerberos サーバーは、IP アドレスではなくホスト名で動作します。Kerberos 認証を使用して PEM サーバーでシングルサインオンを使用するには、DNS レルムを使用してホスト名で次のマシンを構成します。

- Kerberos サーバー
- PEM サーバー PEM Web サーバーおよび PEM バックエンドデータベースサーバー
- クライアントマシン

たとえば、Kerberos サーバーのレルムが `edbpem.org` の場合、Kerberos サーバーホスト名を `Krb5server.edbpem.org` に、PEM サーバーホスト名を `pem.edbpem.org` に、クライアントのホスト名を `pg12.edbpem.org` に設定できます。規則では、DNS ドメイン名をレルムの名前として使用します。

6.3.2.1 1.Kerberos、PEM サーバー、および PEM バックエンドデータベースをインストールします

認証サーバーとして機能するマシンに Kerberos をインストールします。PEM サーバーを別のマシンにインストールします。詳細については、*Installing the PEM server and PEM-HTTPD on separate hosts* を参照してください。

PEM バックエンドデータベース Postgres/ EDB Postgres Advanced Server を PEM サーバーと同じマシンまたは別のマシンにインストールします。詳細については、EDB Docs website のインストール手順を参照してください。

6.3.2.2 2.Kerberos サーバーにプリンシパルを追加する

Apache Web サーバー HTTPD / Apache2 および PEM バックエンドデータベースサーバー PostgreSQL / EDB Postgres Advanced Server の下に展開された PEM Web アプリケーションのプリンシパルを追加します。

```
$ sudo kadmin.local -q "addprinc -randkey HTTP/<HOSTNAME_OF_PEM_SERVER>"
$ sudo kadmin.local -q "addprinc -randkey postgres/<HOSTNAME_OF_PEM_SERVER>"
```

HOSTNAME_OF_PEM_SERVER には、Kerberos サーバーのレルムが含まれている必要があります。たとえば、レルムとして edbpem.org を使用して、PEM サーバーのホスト名として pemdb.edbpem.org を指定できます。

> **注釈**
>
> PEM Web アプリケーションと PEM バックエンドデータベースサーバーが別のマシンにある場合、ホスト名はそれぞれに異なります。

6.3.2.3 3.Kerberos サーバーからキーテーブルを抽出します

PEM Web アプリケーションと PEM バックエンドデータベースサーバーの Kerberos からキーテーブルを抽出します。

```
sudo kadmin.local "ktadd -k <NAME_OF_PEM_WEB_FILE>.keytab HTTP/<HOSTNAME_OF_PEM_
↪SERVER>"
sudo kadmin.local "ktadd -k <NAME_OF_PEM_DB_FILE>.keytab postgres/<HOSTNAME_OF_PEM_
↪SERVER>"
```

キーテーブルを Kerberos サーバーから PEM サーバーにコピーします。

```
scp <NAME_OF_PEM_WEB_FILE>.keytab <OS_USERNAME_ON_PEM_SERVER>@<HOSTNAME_OF_PEM_SERVER>
↪:/tmp
scp <NAME_OF_PEM_DB_FILE>.keytab <OS_USERNAME_ON_PEM_SERVER>@<HOSTNAME_OF_PEM_SERVER>
↪:/tmp
```

PEM サーバーで、キーテーブルを必要な場所に移動し、所有権を変更します。

```
mv /tmp/<NAME_OF_PEM_WEB_FILE>.keytab <PEM_INSTALLATION_DIRECTORY>/share
chown pem <PEM_INSTALLATION_DIRECTORY>/share/<NAME_OF_PEM_WEB_FILE>.keytab
```

```
mv /tmp/<NAME_OF_PEM_DB_FILE>.keytab <DATA_DIRECTORY_OF_POSTGRES>/
chown enterprisedb <DATA_DIRECTORY_OF_POSTGRES>/<NAME_OF_PEM_DB_FILE>.keytab
```

そこで

- NAME_OF_PEM_WEB_FILE は、PEM Web アプリケーションのキーテーブルに指定された名前です。
- NAME_OF_PEM_DB_FILE は、PEM バックエンドデータベースサーバーのキーテーブルに指定された名前です。
- OS_USERNAME_ON_PEM_SERVER は、PEM サーバー上のオペレーティングシステムユーザーの名前です。
- DATA_DIRECTORY_OF_POSTGRES は、インストールされた Postgres データベース PostgreSQL/ EDB Postgres Advanced Server のデータディレクトリのパスです。

6.3.2.4 4.PEM バックエンドデータベースサーバーの構成

postgresql.conf ファイルにキーテーブルの場所を追加します。

```
krb_server_keyfile=FILE:/<DATA_DIRECTORY_OF_POSTGRES>/<NAME_OF_PEM_DB_FILE>.keytab
```

そこで

- NAME_OF_PEM_DB_FILE は、PEM バックエンドデータベースサーバーのキーテーブルに指定された名前です。
- DATA_DIRECTORY_OF_POSTGRES は、インストールされた Postgres データベース PostgreSQL/ EDB Postgres Advanced Server のデータディレクトリのパスです。

krb5.conf ファイルを編集します。

```
$ sudo vim /etc/krb5.conf
[libdefaults]
 default_realm = EDBPEM.ORG
Forwardable = True

[domain_realm]
.edbpem.org = EDBPEM.ORG
edbpem.org = EDBPEM.ORG

[realms]
EDBPEM.ORG = {
   kdc = krb5server.edbpem.org
   admin_server = krb5server.edbpem.org
}
```

データベースサーバーを再起動して、変更を反映します。

```
systemctl restart <POSTGRES_SERVICE_NAME>
```

POSTGRES_SERVICE_NAME は、 Postgres PostgreSQL/ EDB Postgres Advanced Server データベースのサービス名です。たとえば、RHEL または Rocky Linux プラットフォーム上の PostgreSQL 13 データベースの postgresql-13 です。

6.3.2.5 5. 最初のチケットを取得して表示します

kinit ユーティリティは、Kerberos チケットを取得およびキャッシュします。通常、このユーティリティを使用して、チケット交付チケットを取得し、パスワードを入力してキー配布センター KDC から資格情報を復号します。チケット交付チケットは、資格情報キャッシュに保存されます。

klist ユーティリティを使用して、チケットの詳細を表示できます。

> **注釈**
>
> kinit および klist を使用する前に、PEM サーバーとクライアントマシンに Kerberos クライアントをインストールします。

```
$ kinit <USERNAME@REALM>
$ klist
```

これらのコマンドは、プリンシパルと Kerberos チケットを表示します。

> **注釈**
>
> ここで指定される `USERNAME@REALM` は、*pem* データベースに対する pem_admin ロールと CONNECT 特権を持つデータベースユーザーである必要があります。

6.3.2.6 6.PEM サーバーの構成

PEM サーバーで PEMconfigure スクリプトを実行して、Kerberos 認証を使用します。

```
$ sudo PEM_APP_HOST=<HOSTNAME_OF_PEM_SERVER> PEM_KRB_KTNAME=<PEM_INSTALLATION_
↪DIRECTORY/share/<NAME_OF_PEM_WEB_FILE>.keytab <PEM_INSTALLATION_DIRECTORY>/bin/
↪configure-pem-server.sh
```

config_setup.py ファイルで PEM_DB_HOST を構成します。PEM_AUTH_METHOD の値が'kerberos' に設定されていることを確認します。

```
$ sudo vim <PEM_INSTALLATION_DIRECTORY>/share/web/config_setup.py
PEM_DB_HOST=`<HOSTNAME_OF_PEM_SERVER>`
```

.install-config ファイルでホストを構成します。

```
$ sudo vim <PEM_INSTALLATION_DIRECTORY>/share/.install-config
HOST=`<HOSTNAME_OF_PEM_SERVER>`
```

PEM サーバーが Kerberos 認証を使用する場合

- すべての監視対象サーバーはデフォルトで同じ認証を使用します。デフォルトをオーバーライドするには、config_local.py ファイルに、パラメーター ALLOW_DATABASE_CONNECTION_WITHOUT_KERBEROS を追加し、True に設定します。

- 認証されたユーザープリンシパルはすべてレルム USERNAME@REALM が追加され、デフォルトでデータベースユーザー名として渡されます。デフォルトをオーバーライドするには、config_local.py ファイルに、パラメーター PEM_USER_KRB_INCLUDE_REALM を追加し、False に設定します。
- Apache サーバーを再起動します。

```
sudo systemctl restart <SERVICE_NAME>
```

- gss 認証方法を使用するように pg_hba.conf の上部にあるエントリを編集し、データベースサーバーをリロードします。

```
host  pem       +pem_user    <ip_of_pem_server>/32   gss
host  postgres  +pem_user    <ip_of_pem_server>/32   gss
```

```
systemctl reload <POSTGRES_SERVICE_NAME>
```

POSTGRES_SERVICE_NAME は、 Postgres PostgreSQL/ EDB Postgres Advanced Server データベースのサービス名です。たとえば、RHEL または Rocky Linux プラットフォーム上の PostgreSQL 13 データベースの postgresql-13 です。

> **注釈**
>
> PostgreSQL または EDB Postgres Advanced Server 12 以降を使用している場合、接続タイプを `hostgssenc` として指定して、gss 暗号化された接続のみを許可できます。

6.3.2.7 7. ブラウザ設定

Spnego / Kerberos を使用するように PEM Web クライアントにアクセスするようにクライアントマシンのブラウザーを構成します。

Mozilla Firefox の場合

1. about:config ページをロードして、低レベルの Firefox 構成ページを開きます。
2. 検索ボックスに「network.negotiate-auth.trusted-uris 」と入力します。
3. network.negotiate-auth.trusted-uris 設定をダブルクリックし、Kerberos HTTP SPNEGO によって保護される Web サーバーのホスト名またはドメインを入力します。複数のドメインとホスト名はコンマで区切ります。
4. 検索ボックスに「network.negotiate-auth.delegation-uris 」と入力します。
5. network.negotiate-auth.delegation-uris 設定をダブルクリックし、Kerberos HTTP SPNEGO によって保護される Web サーバーのホスト名またはドメインを入力します。複数のドメインとホスト名はコンマで区切ります。
6. **OK** を選択します。

Linux または MacOS での Google Chrome の場合

- google-chrome コマンドに--auth-server-whitelist パラメーターを追加します。たとえば、Linux プロンプトから Chrome を実行するには、次の google-chrome コマンドを使用します。

```
google-chrome --auth-server-whitelist = "hostname/domain"
```

- PEM サーバーを構成すると、ブラウザーで PEM Web インターフェイスにアクセスできます。次の場所に移動します

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

注釈

Postgres クラスターへの接続中に、次のエラーが表示される場合があります。

```
psql -h hostname template1 psql:  GSSAPI continuation error:  Unspecified GSS failure.
Minor code may provide more information GSSAPI continuation error:  Key version is not
available
```

ktutil を使用するか、AD のすべての暗号システムで Postgres キータブを再作成して、暗号化タイプをキータブに追加します。

6.3.3 Configuring the PEM server to use Windows Active Directory domain services for Kerberos authentication (SSPI)

Windows Active Directory ドメインサービスは、IP アドレスではなくホスト名で動作します。Active Directory ドメインサービスを使用して PEM サーバーでシングルサインオンを使用するには、DNS を使用してホスト名で次のマシンを構成します。

・Windows サーバードメインコントローラ

- PEM サーバー PEM Web サーバーおよび PEM バックエンドデータベースサーバー
- クライアントマシン

たとえば、 Windows Active Directory のレルムが edbpem.internal の場合、 Windows サーバーホスト名を Krb5server.edbpem.internal に、PEM サーバーホスト名を pem.edbpem.internal に、クライアントのホスト名を pg12.edbpem.internal に設定できます。

6.3.3.1 1. Active Directory、PEM サーバー、および PEM バックエンドデータベースサーバーのインストール

次のインストールを実行します。

・ 認証サーバーとして機能する Windows サーバードメインコントローラに Active Directory をインストールします。また、Kerberos 認証を使用するように Active Directory ドメインサービスを構成し、起動します。

- PEM サーバーを別の Linux マシンにインストールします。詳細については、*Installing the PEM server and PEM-HTTPD on separate hosts* を参照してください。

- PEM バックエンドデータベース Postgres/ EDB Postgres Advanced Server を PEM サーバーと同じ Linux マシンまたは別の Linux マシンにインストールします。詳細については、EDB Docs website のインストール手順を参照してください。

6.3.3.2 2. サービスプリンシパルとマップする Active Directory にユーザーを作成する

Windows サーバーの Active Directory にユーザーを作成して、PEM Web アプリケーションの HTTP サービスプリンシパルとマップします。

1. **Active Directory ユーザーとコンピューター > <DOMAIN_NAME> > Users** を開きます。右クリックして、 **New > User** を選択します。

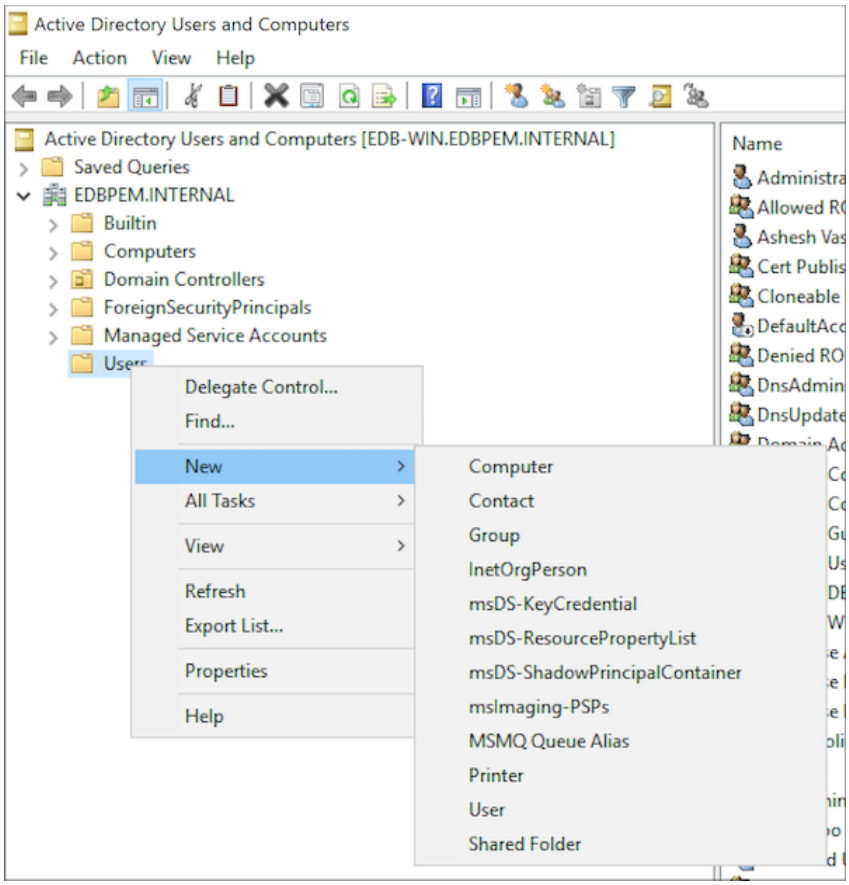


図 1: Active Directory Users and Computers

1. ユーザーの詳細を入力します。

2. パスワードを入力し、**ユーザーは次回ログオンにパスワードを変更する必要があります**をクリアします。また、 **ユーザーはパスワードを変更できません** と**パスワードの有効期限はありません** を選択します。

3. ユーザーの詳細を確認します。

4. [PEMServer Web Properties] ダイアログボックスで、ユーザーを Domain Users グループのメンバーとして追加します。

1. Windows サーバーの Active Directory にユーザー pemserverdb などを作成して、PEM バックエンドデータベースの Postgres サービスプリンシパルとマップします。

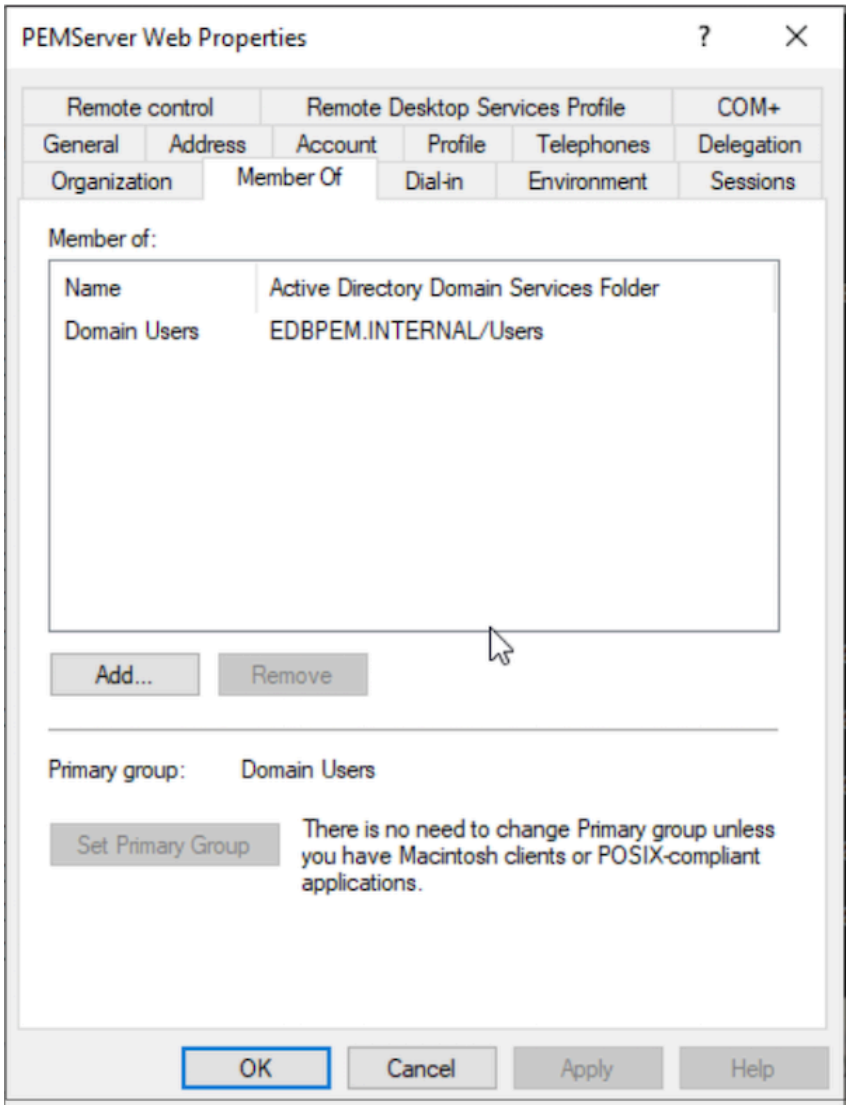


図 2: PEM Server Web Properties

6.3.3.3 3. Active Directory からキーテーブルを抽出する

サービスプリンシパルのキーテーブルを抽出し、作成した各ドメインユーザーにマップします。

1. 管理者として Windows PowerShell を開きます。ユーザー pemserverweb を使用して HTTP サービスプリンシパルマッピングのキーテーブルを作成し、ユーザー pemserverdb を使用して Postgres サービスプリンシパルマッピングのキーテーブルを作成します。

```
ktpass /out pemserver.keytab /princ HTTP/pem.edbpem.internal@EDBPEM.INTERNAL /mapuser␣
↪pemserverweb /crypto AES256-SHA1 +rndpass /target EDBPEM.INTERNAL -ptype KRB5_NT_
↪PRINCIPAL -kvno 0
ktpass /out pemdb.keytab /princ postgres/pem.edbpem.internal@EDBPEM.INTERNAL /mapuser␣
↪pemserverdb /crypto AES256-SHA1 +rndpass /target EDBPEM.INTERNAL -ptype KRB5_NT_
↪PRINCIPAL -kvno 0
```

そこで

- pemserver.keytab は、PEM Web アプリケーションのキーテーブルの名前です。 - pemdb.keytab は、PEM バックエンドデータベースサーバーのキーテーブルの名前です。 - pem.edbpem.internal@EDBPEM.INTERNAL は、PEM サーバーのホスト名です。ここで、@EDBPEM.INTERNAL は@REALM を意味します。- pemserverweb は、PEM Web アプリケーションのユーザーです。 - pemserverdb は、PEM バックエンドデータベースサーバーのユーザーです。 - EDBPEM.INTERNAL は、ターゲットのドメインです。

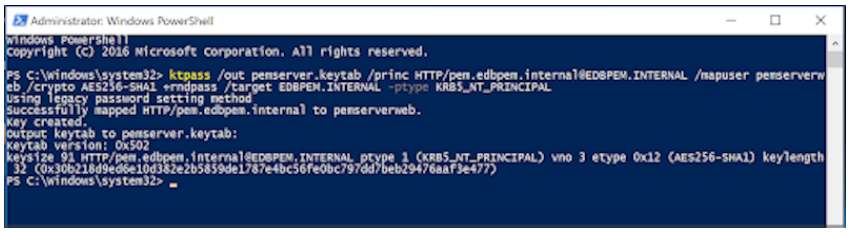


図 3: Windows PowerShell ktpass

> **注釈**
>
> コマンドライン引数`+rndpass` は、ドメインユーザー pemserverweb のパスワードをランダムなパスワードにリセットします。 */target* オプションはオプショナル。

1. [**Accounts**] タブで、ユーザーアカウントの Kerberos サポートを追加します。

> **注釈**
>
> [**Accounts**] タブでは、ユーザーログオン名に**HTTP/pem.edbpem.internal@EDBPEM.INTERNAL**が表示されます。 [**Delegation**] タブは pemserverweb ユーザーに対して有効になっています。

1. [**Delegation**] タブで、作成したユーザーに **Trust this user for delegation to any serviceKerberos Only**) を選択します。

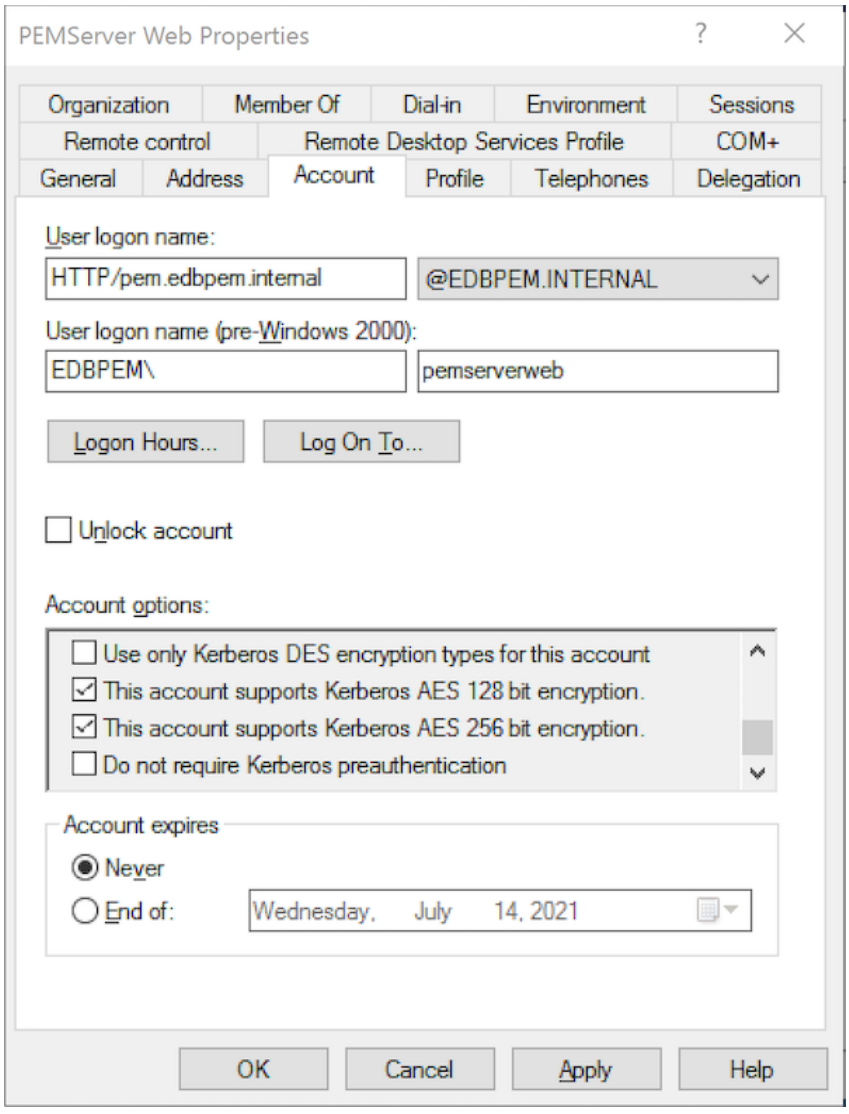


図 4: PEM Server Web Properties

2. 両方のキーテーブルを PEM サーバーホスト、または PEM Web サーバーと PEM バックエンドデータベースサーバーホストにコピーします。別のホストにインストールされている場合。

3. PEM サーバーで、キーテーブルを必要な場所に移動し、所有権を変更します。

```
mv /tmp/pemserver.keytab <PEM_INSTALLATION_DIRECTORY>/share
chown pem <PEM_INSTALLATION_DIRECTORY>/share/pemserver.keytab
```

```
mv /tmp/pemdb.keytab <DATA_DIRECTORY_OF_POSTGRES>/
chown enterprisedb <DATA_DIRECTORY_OF_POSTGRES>/pemdb.keytab
```

そこで

- OS_USERNAME_ON_PEM_SERVER は、PEM サーバー上のオペレーティングシステムユーザーの名前です。 - DATA_DIRECTORY_OF_POSTGRES は、インストールされた Postgres データベース PostgreSQL/ EDB Postgres Advanced Server のデータディレクトリのパスです。

6.3.3.4 4.PEM バックエンドデータベースサーバーの構成

postgresql.conf ファイルにキーテーブルの場所を追加します。

```
krb_server_keyfile=FILE:/<DATA_DIRECTORY_OF_POSTGRES>/pemdb.keytab
```

DATA_DIRECTORY_OF_POSTGRES は、インストールされた Postgres データベース PostgreSQL/ EDB Postgres Advanced Server のデータディレクトリのパスです。

krb5.conf ファイルを編集します。

```
$ sudo vim /etc/krb5.conf
[libdefaults]
default_realm = EDBPEM.INTERNAL
Forwardable = True

[domain_realm]
.edbpem.org = EDBPEM.INTERNAL
edbpem.org = EDBPEM.INTERNAL

[realms]
EDBPEM.INTERNAL = {
  kdc = krb5server.edbpem.internal
  admin_server = krb5server.edbpem.internal
}
```

データベースサーバーを再起動して、変更を反映します。

```
systemctl restart <POSTGRES_SERVICE_NAME>
```

POSTGRES_SERVICE_NAME は、 Postgres PostgreSQL/ EDB Postgres Advanced Server データベースのサービス名です。たとえば、RHEL または Rocky Linux プラットフォーム上の PostgreSQL 13 データベースの

postgresql-13 です。

6.3.3.5 5. 最初のチケットを取得して表示します

kinit ユーティリティは、Kerberos チケットを取得およびキャッシュします。通常、このユーティリティを使用して、キー配布センター KDC から資格情報を復号化するために入力したパスワードを使用して、チケット交付チケットを取得します。チケット交付チケットは、資格情報キャッシュに保存されます。

klist ユーティリティを使用して、チケットの詳細を表示できます。

> **注釈**
>
> kinit および klist を使用する前に、PEM サーバーとクライアントマシンに Kerberos クライアントをインストールする必要があります。

```
$ kinit <USERNAME@REALM>
$ klist
```

これらのコマンドは、プリンシパルと Kerberos チケットを表示します。

> **注釈**
>
> ここで指定される `USERNAME@REALM` は、*pem* データベースに対する pem_admin ロールと CONNECT 特権を持つデータベースユーザーである必要があります。

6.3.3.6 6.PEM サーバーの構成

PEM サーバーで PEMconfigure スクリプトを実行して、Kerberos 認証を使用します。

```
$ sudo PEM_APP_HOST=pem.edbpem.internal PEM_KRB_KTNAME=<PEM_INSTALLATION_DIRECTORY/
↪share/pemserver.keytab <PEM_INSTALLATION_DIRECTORY>/bin/configure-pem-server.sh
```

config_setup.py ファイルで、PEM_DB_HOST を構成し、PEM_AUTH_METHOD の値が'kerberos' に設定されていることを確認します。

```
$ sudo vim <PEM_INSTALLATION_DIRECTORY>/share/web/config_setup.py
PEM_DB_HOST=`pem.edbpem.internal`
```

.install-config ファイルで HOST を構成します。

```
$ sudo vim <PEM_INSTALLATION_DIRECTORY>/share/.install-config
HOST=`pem.edbpem.internal`
```

PEM サーバーが Kerberos 認証を使用する場合

- すべての監視対象サーバーはデフォルトで同じ認証を使用します。デフォルトをオーバーライドするには、config_local.py ファイルに、パラメーター ALLOW_DATABASE_CONNECTION_WITHOUT_KERBEROS を追加し、True に設定します。

- 認証されたユーザープリンシパルはすべてレルム USERNAME@REALM が追加され、デフォルトでデータベースユーザー名として渡されます。デフォルトをオーバーライドするには、config_local.py ファイルに、パラメーター PEM_USER_KRB_INCLUDE_REALM を追加し、False に設定します。

Apache サーバーを再起動します。

```
sudo systemctl restart <SERVICE_NAME>
```

gss 認証方法を使用するように pg_hba.conf の上部にあるエントリを編集します。次に、データベースサーバーをリロードします。

= ```shell host pem +pem_user /32 gss host postgres +pem_user /32 gss

``````

```
systemctl reload <POSTGRES_SERVICE_NAME>
``` `


`POSTGRES_SERVICE_NAME` は、 Postgres PostgreSQL/ EDB Postgres Advanced Server データベースのサービス名です。たとえば、RHEL または Rocky Linux プラットフォーム上の PostgreSQL 13 データベースの `postgresql-13` です。

..  Note::
接続タイプを `hostgssenc` として指定することはできません。 Windows は、gss 暗号化された接続をサポートしていません。

##  7. ブラウザ設定

Spnego / Kerberos を使用するように PEM Web クライアントにアクセスするようにクライアントマシンのブラウザーを構成します。

Mozilla Firefox の場合

- `about:config` ページをロードして、低レベルの Firefox 構成ページを開きます。

- 検索ボックスに「`network.negotiate-auth.trusted-uris` 」と入力します。

- `network.negotiate-auth.trusted-uris` 設定をダブルクリックし、Kerberos HTTP SPNEGO によって保護された Web サーバーのホスト名またはドメインを入力します。複数のドメインとホスト名はコンマで区切ります。

- 検索ボックスに「`network.negotiate-auth.delegation-uris` 」と入力します。

- `network.negotiate-auth.delegation-uris` 設定をダブルクリックし、Kerberos HTTP SPNEGO によって保護された Web サーバーのホスト名またはドメインを入力します。複数のドメインとホスト名はコンマで区切ります。

-  **OK** を選択します。

(次のページに続く)
```

(前のページからの続き)

```
Linux または MacOS での Google Chrome の場合

- `google-chrome` コマンドに `--auth-server-whitelist` パラメーターを追加します。たとえば、
Linux プロンプトから Chrome を実行するには、次の `google-chrome` コマンドを使用します。

```ini
  google-chrome --auth-server-whitelist = "hostname/domain"
```

- PEM サーバーを構成すると、ブラウザーで PEM Web インターフェイスにアクセスできます。次の場所に移動します

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

**注釈**

Postgres クラスターへの接続中に、次のエラーが表示される場合があります。

```
psql -h hostname template1 psql:  GSSAPI continuation error:  Unspecified GSS failure.
Minor code may provide more information GSSAPI continuation error:  Key version is not
available
```

ktutil を使用するか、AD のすべての暗号システムで Postgres キータブを再作成して、暗号化タイプをキータブに追加します。

## 6.4 Connection pooling using PgBouncer

Windows 以外のシステムで PEM エージェントとサーバー間の接続を制限するには、接続プーラーとして PgBouncer を実装できます。このプロセスには、3 つの主要なフェーズが含まれます。

- ** *Preparing the PEM database server* **

PgBouncer で使用する PEM データベースサーバーを準備する手順を提供します。

- ** *Configuring PgBouncer* **

PEM データベースサーバーと連携できるように PgBouncer の構成に関する詳細情報を提供します。

- ** *Configuring the PEM agent* **

PgBouncer に接続するように PEM エージェントを構成することに関する詳細情報を提供します。

PEM Web インターフェイスの使用の詳細については、*Accessing the web interface* を参照してください。
```

### 6.4.1 Preparing the PEM database server

PgBouncer で PEM の接続プーリングを有効にするには、専用ユーザーを構成し、PEM データベースサーバーで SSL キーと証明書を作成する必要があります。

この例では、EDB Postgres Advanced Server バージョン 16 を使用する RHEL ベースのオペレーティングシステムで、enterprisedb ユーザーを使用して PEM データベースサーバーを準備する方法を示しています。データの場所、構成ファイルとキーファイル、および構成の実行に使用するユーザー OS および Postgres ディストリビューションによって異なる場合があります。

#### 6.4.1.1 前提条件

- PEM データベースサーバーの pem データベースに接続します。
- Postgres ディストリビューションに基づいて、enterprisedb または postgres ユーザーとして接続します。

| Postgres distribution | User |
|---|---|
| EDB Postgres Advanced Server | enterprisedb |
| EDB Postgres Extended Server | postgres |
| PostgreSQL | postgres |

#### 6.4.1.2 PgBouncer-PEM 接続のユーザーとロールの作成

1. pem_agent_pool メンバーシップを使用して pgbouncer という名前の専用ユーザーを作成します。このユーザーは、すべてのエージェントデータベースクエリを転送することにより、PgBouncer から PEM データベースへの接続を提供します。

```
CREATE ROLE pgbouncer PASSWORD ANY_PASSWORD LOGIN;
__OUTPUT__
CREATE ROLE
```

```
GRANT pem_agent_pool TO pgbouncer;
__OUTPUT__
GRANT ROLE
```

1. pem_admin および pem_agent_pool ロールメンバーシップを使用して pem_admin1 スーパーユーザーではないという名前のユーザーを作成します。このユーザーはエージェントを PEM サーバーに登録し、PEM データベースへのアクセスを管理します。

```
CREATE ROLE pem_admin1 PASSWORD ANY_PASSWORD LOGIN CREATEROLE;
__OUTPUT__
CREATE ROLE
```

```
GRANT pem_agent_pool TO pem_admin1;
__OUTPUT__
GRANT ROLE
```

```
GRANT pem_agent TO pem_admin1 WITH ADMIN OPTION;
__OUTPUT__
GRANT ROLE
```

1. pgbouncer ユーザーに CONNECT 特権を付与します。

```
GRANT CONNECT ON DATABASE pem TO pgbouncer;
__OUTPUT__
GRANT
```

1. pem スキーマの pgbouncer ユーザーに USAGE 特権を付与します。

```
GRANT USAGE ON SCHEMA pem TO pgbouncer;
__OUTPUT__
GRANT
```

1. pem.get_agent_pool_auth(text) ファンクションで pgbouncer ユーザーに EXECUTE 特権を付与します。例

```
GRANT EXECUTE ON FUNCTION pem.get_agent_pool_auth(text) TO pgbouncer;
__OUTPUT__
GRANT
```

1. pem.create_proxy_agent_user(varchar) ファンクションを使用して、pem_agent_user1 という名前のユーザーを作成します。このプロキシユーザーは、すべてのエージェントと PgBouncer 間の接続を提供します。

```
SELECT pem.create_proxy_agent_user(pem_agent_user1);
__OUTPUT__
create_proxy_agent_user
-------------------------
(1 row)
```

このファンクションは、同じ名前とランダムなパスワードを持つユーザーを作成し、pem_agent および pem_agent_pool ロールをユーザーに付与します。このアプローチにより、PgBouncer は、エージェントに代わってプロキシユーザーを使用できます。

#### 6.4.1.3 構成ファイルを更新して PgBouncer-PEM 接続を許可する

1. pgbouncer ユーザーが、SSL 認証方法を使用して pem データベースに接続できるようにします。これを行うには、PEM データベースサーバーの pg_hba.conf ファイルに hostssl pem エントリーを追加します。

ルールのリストで、+pem_agent ユーザーに割り当てられた他のルールの前に hostssl pem エントリーを配置してください。

```
# Allow the PEM agent proxy user (used by pgbouncer)
# to connect the to PEM server using SSL

hostssl pem     +pem_agent_pool 127.0.0.1/32  cert map=pem_agent_pool
```

1. 次の行を$PGDATA/pg_ident.conf ユーザーマッピングファイルに追加することにより、PEM サーバーが PgBouncer-PEM 接続に関係するすべてのユーザーをマップできるようにします。

```
pem_agent_pool  pem_agent_pool  pem_agent_user1
pem_agent_pool  pem_agent_pool  pem_admin1
pem_agent_pool  pem_agent_pool  pgbouncer
```

1. Postgres サービスを再起動します。 <postgres_service> を Postgres インスタンスの systemd サービス名の名前に置き換えます。

```
systemctl restart <postgres_service>
```

#### 6.4.1.4 PgBouncer-PEM 認証の SSL キーと証明書の作成

pem_agent_pool グループロールのキーと証明書を作成します。次に、ファイルを PgBouncer インスタンスに移動して、PEM データベースサーバーと PgBouncer 間の認証を許可します。

この例では、RHEL で EDB Postgres Advanced Server を実行します。環境変数を設定するときは、オペレーティングシステムと Postgres ディストリビューションに応じて正しいディレクトリを選択します。

1. $DATA_DIR 環境変数をデータディレクトリに設定します。

```
export DATA_DIR=/var/lib/edb/as16/data
```

OS および Postgres バージョンごとのデータディレクトリ

次に、オペレーティングシステムおよび Postgres バージョンごとの他のデフォルトデータディレクトリの例をいくつか示します。

| Postgres version | RHEL/Rocky Linux/AlmaLinux/SLES | Debian/Ubuntu |
|---|---|---|
| EDB Postgres Advanced Server 16 | /var/lib/edb/as16/data | /var/lib/edb-as/16/main |
| EDB Postgres Extended Server 16 | /var/lib/edb/edb-pge/16/data | /var/lib/edb-pge/16/main |
| PostgreSQL 16 | /var/lib/edb/pgsql/16/data | /etc/postgresql/16/main |

1. $USER_HOME 環境変数を、ユーザーがアクセスできるホームディレクトリに設定します。

```
export USER_HOME=/var/lib/edb
```

OS および Postgres バージョンごとのユーザーホームディレクトリ

次に、オペレーティングシステムおよび Postgres バージョンごとの他のデフォルトのホームディレクトリの例をいくつか示します。

| Postgres version | RHEL/Rocky Linux/AlmaLinux/SLES | Debian/Ubuntu |
|---|---|---|
| EDB Postgres Advanced Server 16 | /var/lib/edb | /var/lib/edb-as |
| EDB Postgres Extended Server 16 | /var/lib/pgsql | /var/lib/postgresql |
| PostgreSQL 16 | /var/lib/pgsql | /var/lib/postgresql |

1. openssl で署名キーを作成します。

```
openssl genrsa -out pem_agent_pool.key 4096
```

1. 証明書署名要求 CSR を作成します。必要に応じて、<...> の-subj 属性を置き換えます。共通名 CN が pem_agent_pool グループロール名に設定されていることを確認します。

```
openssl req -new -key pem_agent_pool.key -out pem_agent_pool.csr -subj /C=<COUNTRY>/
↪ST=<STATE>/L=<LOCATION>/O=<ORGANISATION>/CN=pem_agent_pool
```

1. PEM CA とキーを使用して CSR に署名します。

```
openssl x509 -req -days 365 -in pem_agent_pool.csr -CA $DATA_DIR/ca_certificate.crt -
↪CAkey $DATA_DIR/ca_key.key -CAcreateserial -out pem_agent_pool.crt
```

1. 作成したキーと証明書を、enterprisedb ユーザーがアクセスできるパスに移動します。

この例では、enterprisedb ユーザーのホームディレクトリに~/.postgresql フォルダーを作成し、権限があることを確認します。

```
mkdir -p $USER_HOME/.postgresql
mv pem_agent_pool.key pem_agent_pool.crt $USER_HOME/.postgresql
chmod 0600 $USER_HOME/.postgresql/pem_agent_pool.key
chmod 0644 $USER_HOME/.postgresql/pem_agent_pool.crt
chown enterprisedb:enterprisedb $USER_HOME/.postgresql/pem_agent_pool.*
```

### 6.4.2 Configuring PgBouncer

PEM データベースサーバーと連携するように PgBouncer を構成する必要があります。

#### 6.4.2.1 前提条件

- EDB Postgres Advanced Server を実行している場合、EDB PgBouncer をインストールします。
- EDB Postgres Extended Server または PostgreSQL を実行している場合、コミュニティ PgBouncer をインストールします。

#### EDB PgBouncer および PgBouncer インストールの考慮事項

構成手順でのディレクトリとファイルの名前と場所、およびユーザーは、PgBouncer のコミュニティバージョンと EDB PgBouncer のどちらをインストールしたかによって異なります。コミュニティ PgBouncer をインストールした場合コミュニティリポジトリと EDB リポジトリからインストールするかにかかわらず、作業例のファイルとディレクトリの名前を PgBouncer の値に置き換えます。

| Name | PgBouncer | EDB PgBouncer |
|---|---|---|
| PgBouncer directory | */etc/pgbouncer* | */etc/edb/pgbouncer<1.x>* |
| ini file | *pgbouncer.ini* | *edb-pgbouncer.ini* |
| HBA file | */etc/pgbouncer/hba_file* | */etc/edb/pgbouncer<1.x>/hba_file* |
| Service file | *pgbouncer* | *edb-pgbouncer-<1.x>* |
| User | *postgres* | *enterprisedb* |

### 6.4.2.2 PgBouncer の構成

この例では、enterprisedb システムユーザーを使用して EDB PgBouncer を構成します。

コミュニティ PgBouncer を実行している場合は、*EDB PgBouncer および PgBouncer インストールの考慮事項* の説明に従って、ディレクトリ、ファイル、およびユーザーの名前を置き換えます。

1. ターミナルウィンドウを開き、PgBouncer ディレクトリに移動します。
2. PgBouncer の `etc` ディレクトリ `edb-pgbouncer.ini` が存在する場所の所有者を `enterprisedb` に変更し、ディレクトリのアクセス許可を `0700` に変更します。

```
chown -R enterprisedb:enterprisedb /etc/edb/pgbouncer<1.x>
chmod 0700 /etc/edb/pgbouncer<1.x>
```

1. `edb-pgbouncer.ini` ファイルの内容を変更します。

```
  [databases]
  ;; Change the pool_size according to maximum connections allowed
  ;; to the PEM database server as required.
  ;; auth_user will be used for authenticate the db user (proxy
  ;; agent user in our case)
  pem = port=5444 host=127.0.0.1 dbname=pem auth_user=pgbouncer pool_size=80 pool_
↪mode=transaction
- = port=5444 host=127.0.0.1 dbname=pem auth_user=pgbouncer pool_size=10

  [pgbouncer]
  logfile = /var/log/edb/pgbouncer<1.x>/edb-pgbouncer-<1.x>.log
  pidfile = /var/run/edb/pgbouncer<1.x>/edb-pgbouncer-<1.x>.pid
  listen_addr = *
  ;; Agent needs to use this port to connect the pem database now
  listen_port = 6432
  ;; Set to require to ensure SSL certificates are used for connections
```

(次のページに続く)

(前のページからの続き)

```
;; for PEM Agents
client_tls_sslmode = require
;; These are the root.crt, server.key, server.crt files present
;; in the present under the data directory of the PEM database
;; server, used by the PEM Agents for connections.
client_tls_ca_file = /var/lib/edb/as16/data/root.crt
client_tls_key_file = /var/lib/edb/as16/data/server.key
client_tls_cert_file = /var/lib/edb/as16/data/server.crt
;; Allow pgBouncer to use pem_agent_pool certificate
;; and key for connections to the server.
server_tls_key_file = /var/lib/edb/.postgresql/pem_agent_pool.key
server_tls_cert_file = /var/lib/edb/.postgresql/pem_agent_pool.crt
;; Use hba file for client connections
auth_type = hba
;; HBA file
auth_hba_file = /etc/edb/pgbouncer<1.x>/hba_file
;; Use pem.get_agent_pool_auth(TEXT) function to authenticate
;; the db user (used as a proxy agent user).
auth_query = SELECT * FROM pem.get_agent_pool_auth($1)
;; DB User for administration of the pgbouncer
admin_users = pem_admin1
;; auth_dbname and auth_user allow
;; admin console login by admin_users and stats_users
auth_dbname = pem
auth_user = pgbouncer
;; DB User for collecting the statistics of pgbouncer
stats_users = pem_admin1
server_reset_query = DISCARD ALL
;; Change based on the number of agents installed/required
max_client_conn = 500
;; Close server connection if its not been used in this time.
;; Allows to clean unnecessary connections from pool after peak.
server_idle_timeout = 60
```

**!!!note *auth_user* の詳細については、 Authentication settings を参照してください。 .. ::**

1. 次のコンテンツを含む PgBouncer の HBA ファイル */etc/edb/pgbouncer<1.x>/hba_file* を作成します。

```
# Use the authentication method scram-sha-256 for local connections
# between the pem database & the pgbouncer (virtual) database.
local pgbouncer all scram-sha-256
# Use the authentication method scram-sha-256 for remote connections
# to pgbouncer (virtual database) using the enterprisedb user.
host pgbouncer,pem pem_admin1 0.0.0.0/0 scram-sha-256
# Use the authentication method cert for TCP/IP connections
# to the pem database using pem_agent_user1
```

(次のページに続く)

(前のページからの続き)

```
hostssl pem pem_agent_user1 0.0.0.0/0 cert
```

1. HBA ファイル (/etc/edb/pgbouncer<1.x>/hba_file) の所有者を enterprisedb に変更し、ディレクトリ権限を 0600 に変更します。

```
chown enterprisedb:enterprisedb /etc/edb/pgbouncer<1.x>/hba_file
chmod 0600 /etc/edb/pgbouncer<1.x>/hba_file
```

1. PgBouncer サービスを有効にし、サービスを開始します。

```
systemctl enable edb-pgbouncer-<1.x>
__OUTPUT__
Created symlink from
/etc/systemd/system/multi-user.target.wants/edb-pgbouncer-<1.x>.service
to /usr/lib/systemd/system/edb-pgbouncer-<1.x>.service.
```

```
systemctl start edb-pgbouncer-<1.x>
```

### 6.4.3 Configuring the PEM agent

#### 6.4.3.1 前提条件

Install the PEM agent 。

!!!note PgBouncer を介して接続するように agent.cfg ファイルで true に設定された enable_smtp 、enable_snmp 、または enable_webhook を使用して PEM エージェントを構成しないでください。SNMP、SMTP、および Webhook スプーラーは、Postgres が提供する LISTEN/NOTIFY メカニズムを使用して、通知を非同期に送信します。PgBouncer はトランザクションモードでの LISTEN/NOTIFY メカニズムをサポートしていないため、エージェントを PgBouncer に接続すると、通知が遅延されるか、まったく配信されない場合があります。代わりに、PEM エージェントを PEM バックエンドデータベースに直接接続します。

#### 6.4.3.2 新しい PEM エージェントの構成

PEM エージェントをインストールした後、特定の PEM データベースサーバーと連携するように構成します。

```
PGSSLMODE=require PEM_SERVER_PASSWORD=pem_admin1_password \
      /usr/edb/pem/agent/bin/pemworker \
      --register-agent \
      --pem-server 172.16.254.22 \
      --pem-port 6432 \
      --pem-user pem_admin1 \
      --pem-agent-user pem_agent_user1 \
      --display-name  *Agent_Name*  \
__OUTPUT__
Postgres Enterprise Manager Agent registered successfully!
```

このコマンドでは、 --pem-agent-user 引数は、pem_agent_user1 データベースユーザーの SSL 証明書とキーペアを/root/.pem ディレクトリに作成するようにエージェントに指示します。

例

```
/root/.pem/pem_agent_user1.crt

/root/.pem/pem_agent_user1.key
```

PEM エージェントは、キーを使用して PEM データベースサーバーに pem_agent_user1 として接続します。また、/usr/edb/pem/agent/etc/agent.cfg という名前のエージェント構成ファイルも作成します。

使用する agent-user を記載した行が agent.cfg 構成ファイルにあります。例

```
cat /usr/edb/pem/agent/etc/agent.cfg
[PEM/agent]
pem_host=172.16.254.22
pem_port=6432
agent_id=12
agent_user=pem_agent_user1
agent_ssl_key=/root/.pem/pem_agent_user1.key
agent_ssl_crt=/root/.pem/pem_agent_user1.crt
log_level=warning
log_location=/var/log/pem/worker.log
agent_log_location=/var/log/pem/agent.log
long_wait=30
short_wait=10
alert_threads=0
enable_smtp=false
enable_snmp=false
enable_webhook=false
max_webhook_retries=3
allow_server_restart=true
allow_package_management=false
allow_streaming_replication=false
max_connections=0
connect_timeout=-1
connection_lifetime=0
allow_batch_probes=false
heartbeat_connection=false
```

#### 6.4.3.3 既存の PEM エージェントの構成

既存の PEM エージェントを使用している場合、SSL 証明書とキーファイルをターゲットマシンにコピーし、ファイルを再利用できます。ファイルを変更し、新しいパラメーターを追加して、既存の agent.cfg ファイル内のいくつかのパラメーターを置き換える必要があります。

1. agent_user をエージェントとして使用する行を追加します。

```
agent_user=pem_agent_user1
```

1. ポートを更新して PgBouncer ポートを指定します。

```
pem_port=6432
```

1. 証明書とキーパスの場所を更新します。

```
agent_ssl_key=/root/.pem/pem_agent_user1.key
agent_ssl_crt=/root/.pem/pem_agent_user1.crt
```

別の方法として、エージェント自己登録スクリプトを実行できます。ただし、そのプロセスは新しいエージェント ID を作成します。エージェント自己登録スクリプトを実行する場合、新しいエージェント ID を既存の ID に置き換え、pem.agent テーブルの新しいエージェント ID のエントリを無効にする必要があります。例

```
pem=# UPDATE pem.agent SET active = false WHERE id = <new_agent_id>;
__OUTPUT__

UPDATE 1
```

**注釈**

既存の SSL 証明書、キーファイル、およびエージェント構成ファイルのバックアップを保存します。

## 6.5 Upgrade and migration

PEM インストール、バックエンドデータベース、SQL プロファイラーを含む Postgres Enterprise ManagerPEM コンポーネントをアップグレードできます。PEM サーバーを移動することもできます。

### 6.5.1 Upgrading a PEM installation

PEM インストールをアップグレードするプロセスはプラットフォーム固有です。Linux プラットフォームではネイティブパッケージを使用し、Windows プラットフォームでは Windows で使用可能な PEM グラフィカルインストーラーを使用して PEM エージェントまたはサーバーを更新できます。

PEM インストーラーと RPM へのリンクは、 EDB website で入手できます。

#### 6.5.1.1 Upgrading a PEM installation on a Linux host

Linux ホストで PEM コンポーネントソフトウェアをアップグレードするには、次の順序で PEM コンポーネントパッケージの新しいバージョンをインストールします。

1. PEM サーバーホストを除く各監視対象ノードで PEM エージェントパッケージインストーラーを起動します。

2. PEM サーバーパッケージインストーラーを起動します。PEM サーバーと、PEM サーバーホストにある PEM エージェントの両方をアップグレードします。Web アプリケーションとバックエンドデータベースが別のホストで実行されている場合、最初にデータベースホストを更新します。

インストール中に、コンポーネントのインストールは既存のインストールを検出し、アップグレードを実行します。PEM エージェントとサーバーをアップグレードした後、必要に応じて SQL プロファイラーをアップグレードできます。この手順はプラットフォーム固有です。

### PEM サーバーインストールのアップグレード

隔離されたネットワーク内のマシンにインストールされている PEM サーバーをアップグレードする場合は、PEM サーバーをアップグレードする前にそのマシンに PEM リポジトリを作成する必要があります。詳細については、*Creating an EDB repository on an isolated network* を参照してください。

PEM サーバーインストールをアップグレードするには

```
sudo <package-manager> upgrade edb-pem
```

<package-manager> は、オペレーティングシステムで使用されるパッケージマネージャーです。

| Package manager | Operating system |
|---|---|
| dnf | RHEL 8/9 and derivatives |
| zypper | SLES |
| apt-get | Debian and Ubuntu |

RHEL 同様のシステムで PEM 9 からアップグレードする場合、 --allowerasing オプションを含めて、古い edb-pem-docs パッケージによってアップグレードがブロックされないようにする必要があります。

```
Configuration file /lib/systemd/system/pemagent.service
==> File on system created by you or by a script.
==> File also in package provided by package maintainer.
  What would you like to do about it?  Your options are:
    Y or I  : install the package maintainers version
    N or O  : keep your currently-installed version
      D     : show the differences between the versions
      Z     : start a shell to examine the situation
 The default action is to keep your current version.
 ***  pemagent.service (Y/I/N/O/D/Z) [default=N] ?
```

- 無人アップグレードを実行するには、 apt オプションを使用して決定を強制できます。- **新しいパッケージバージョンをインストールするには**コマンドに o Dpkg::Options::="--force-confnew" を追加します。 - 例 apt -o Dpkg::Options::="--force-confnew" install edb-pem - **既存のバージョンを保持するには**コマンドに o Dpkg::Options::="--force-confdef" -o Dpkg::Options::="--force-confold" を追加します。 - 例 apt -o Dpkg::Options::="--force-confdef" -o Dpkg::Options::="--force-confold" install edb-pem PEM サーバーをアップグレードした後、PEM サーバーを構成する必要があります。詳細については、*Configuring the PEM server to use two-factor authentication* を参照してください。

> **注釈**
>
> PEM バックエンドデータベースサーバーと PEM サーバーをアップグレードする場合は、configure スクリプトを実行する前に、*/usr/edb/pem/share/.install-config* ファイルの新しいバージョンを指す `PG_INSTALL_PATH` および `DB_UNIT_FILE` パラメーターを更新します。

### PEM サーバーの構成

PEM サーバーをアップグレードした後、次のコマンドを使用して PEM サーバーを構成できます。

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

configure スクリプトは、スクリプトの実行中に古い PEM サーバー構成ファイルの値を使用します。詳細については、

Configuring the PEM server on Linux platforms を参照してください。

> **注釈**
>
> -configure スクリプトは、アップグレードプロセスの後にのみスーパーユーザーパスワードを必要とします。

- configure スクリプトがスタックした場合は、`alert_threads>0` で PEM エージェントを停止します。このようなエージェントの詳細を取得するには、次のクエリを実行します。

```
SELECT agent_id FROM pem.agent_config WHERE param=alert_threads AND value > 0;
```

実行中のエージェントを停止し、configure スクリプトを再実行します。

問題が解決しない場合は、クエリを実行して、スタックしたアラートプロセスを終了します。

```
SELECT pg_terminate_backend(pid) FROM pg_stat_activity WHERE query=SELECT pem.process_
↪one_alert();
```

次に、configure スクリプトを再実行します。

### PEM エージェントインストールのアップグレード

PEM エージェントをアップグレードするには

```
sudo <package-manager> upgrade edb-pem-agent
```

<package-manager> は、オペレーティングシステムで使用されるパッケージマネージャーです。

| Package manager | Operating system |
|---|---|
| dnf | RHEL 8/9 and derivatives |
| zypper | SLES |

次のページに続く

表 13 – 前のページからの続き

| Package manager | Operating system |
|---|---|
| apt-get | Debian and Ubuntu |

#### 6.5.1.2 Upgrading a PEM installation on a Windows host

Windows ホストで PEM コンポーネントソフトウェアをアップグレードするには、次の順序で PEM コンポーネントインストーラーの新しいバージョンを呼び出します。

1. PEM サーバーホストを除く各監視対象ノードで PEM エージェントインストーラーを起動します。
2. PEM サーバーインストーラーを起動します。このインストーラーは、PEM サーバーと、PEM サーバーホストにある PEM エージェントの両方をアップグレードします。

インストール中に、コンポーネントインストーラーは既存のインストールを自動的に検出し、アップグレードを実行します。PEM エージェントとサーバーをアップグレードした後、必要に応じて SQL プロファイラーをアップグレードできます。この手順はプラットフォーム固有です。

##### Windows ホストでの PEM エージェントのアップグレード

現在 PEM エージェントによって監視されているシステムをより新しい PEM エージェントにアップグレードするには、エージェントが監視しているシステムで PEM エージェントインストーラーの新しいバージョンをダウンロードして起動します。

1. インストーラーを起動するには、ダウンロードしたインストーラーアイコンを右クリックし、**Run as Administrator** を選択します。 PEM Agent Setup ウィザードが開きます。
2. ライセンス契約を読んで同意し、 **Next** を選択します。
3. セットアップウィザードは、既存のエージェントを自動的に検出し、インストールされたバージョンをアップグレードします。**Next** を選択します。
4. pemAgent Service Account ダイアログボックスは、PEM エージェントサービスを実行するアカウントのパスワードの入力を求められる場合があります。プロンプトが表示されたら、パスワードを入力し、**Next** を選択します。
5. インストールが間もなく開始されることを通知する [ Ready to install ] ダイアログボックスは、[ **Next** ] を選択します。ウィザードは PEM エージェントを最新バージョンにアップグレードします。
6. PEM Agent Setup ウィザードは、インストールが完了すると通知します。**Finish** を選択します。
7. インストールが完了すると、マシンを再起動するように求められます。**Yes** を選択して、マシンと PEM エージェントを再起動します。

##### Windows ホストでの PEM サーバーのアップグレード

PEM サーバーインストーラーを使用すると、PEM サーバーのメジャーバージョン間でアップグレードできます。

1. インストーラーを起動するには、ダウンロードしたインストーラーを右クリックし、**Run as Administrator** を選択します。

2. PEM Server Setup ウィザードがあなたを迎えます。 **Next** を選択します。

3. PEM Server Setup ウィザードは、ライセンス契約に同意するように求めます。ライセンス契約を確認した後、**I accept the Agreement** を選択し、 **Next** を選択します。

4. ウィザードは、PEM サーバーホストで既存の PEM サーバーインストールを確認します。ウィザードはインストールを見つけると、アップグレードを実行します。**Next** を選択します。

PEM サーバーをアップグレードする前に、ウィザードは、新しい PEM サーバーの要件が存在することを確認します。サポートコンポーネントが欠落しているか、新しい PEM インストールをサポートしていないバージョンである場合、ウィザードは依存関係をアップグレードする必要があることを通知します。次に、必要なインストーラーを呼び出します。

1. インストールウィザードが依存関係のアップグレードを完了すると、マシンを再起動するように求められます。**No** を選択して、アップグレードプロセスを続行します。

2. ウィザードは [ Database Server Installation Details ] ダイアログボックスを開き、PEM バックエンドデータベースのデータベーススーパーユーザーの接続資格情報の入力を求めます。提供する

- **User** フィールドのデータベーススーパーユーザーの名前。
- **Password** フィールドのデータベーススーパーユーザーに関連付けられたパスワード。

**Next** を選択します。

1. pemAgent Service Account ダイアログボックスは、PEM エージェントサービスを実行するアカウントのパスワードの入力を求められる場合があります。プロンプトが表示されたら、パスワードを入力し、**Next** を選択します。

2. Ready to Install ダイアログボックスは、セットアップウィザードがインストールを実行する準備ができたことを示します。**Next** を選択してインストールを開始します。

PEM サーバーおよび PEM サーバーと同じホストにあるエージェントをアップグレードし、Web サービスを構成すると、PEM セットアップウィザードは、サービスがリッスンしているポートを通知します。PEM クライアントを使用して PEM サーバーに接続するときにこのポート番号を使用します。

1. **OK** を選択します。PEM サーバーセットアップウィザードは、インストールが完了したことを通知します。

2. マシンを再起動するようにプロンプトが表示されたら、 **Yes** を選択してマシンと httpd サービスを再起動します。

PEM バックエンドデータベースサーバーと PEM-HTTPD を別のホストにインストールした場合、PEM サーバーインストーラーを各ホストで 1 回ずつ、2 回実行する必要があります。言語パックインストーラーを抽出し、PEM インストーラーを呼び出す前に PEM-HTTPD のホストにインストールします。インストーラーを呼び出して言語パックを抽出するときに、次のキーワードを含めます。

```
--extract-languagepack <path>
```

<path> は、言語パックインストーラーを抽出するための既存のパスを指定します。

> **注釈**
>
> デフォルトでは、EDB Language Pack は `C:edblanguagepackv1` にインストールされます。

StackBuilder Plus を使用して PEM サーバーをアップグレードしている場合、エラーが表示される場合があります。エラーが表示された後、PEM はインストールが完了したことを報告します。ただし、インストールは完了していません。ダウンロードした場所からインストーラーファイルを呼び出して、インストールを行う必要があります。

PEM サーバーをアップグレードした後、バックエンドデータベースをより新しいバージョンにアップグレードすることができます。詳細については、*Upgrading the PEM backend Postgres database* を参照してください。

#### 6.5.1.3 Upgrading an HA PEM installation

*high availability PEM* を使用している場合、PEM インストールのメジャーまたはマイナーアップグレードを実行できます。

提供されているコマンド例は、RHEL ベースのオペレーティングシステムを前提としています。

!!!重要このアップグレード手順は、PEM クラスターのダウンタイムを最小限に抑えるように設計されています。ただし、プライマリデータベースインスタンスを 1 回再起動します。プライマリを再起動し、昇格するレプリカの 1 つを回避するには、自動フェイルオーバーが無効になっていることを確認します。

Patroni を使用している場合、 `/usr/edb/pem/share/.install-config` の `DB_UNIT_FILE` の値が `patroni` に設定されていることを確認します。 *installation instructions* で詳細に説明します。

##### プライマリをアップグレードする

プライマリサーバーで

1. PEM パッケージを更新します。

```
dnf upgrade edb-pem --allowerasing
```

1. アップグレードスクリプトを実行します。

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

##### レプリカをアップグレードする

各レプリカサーバーで

1. PEM パッケージを更新します。

```
dnf upgrade edb-pem --allowerasing
```

1. PEM 9 または PEM 10.0 から PEM 10.x に更新する場合は、libpq5 がインストールされていることを確認します。これは `configure-pem-server.sh` によってプライマリで実行されます。

```
dnf install libpq5
```

1. 次の構成ファイルをプライマリからコピーします。

```
/etc/httpd/conf.d/edb-pem.conf
/etc/httpd/conf.d/edb-ssl-pem.conf
/usr/edb/pem/share/.install-config
/usr/edb/pem/web/pem.wsgi
/usr/edb/pem/web/config_setup.py
```

1. Web サーバーとして NGINX を使用している場合、次のファイルもプライマリからコピーします。

```
/etc/edb-uwsgi/uwsgi.ini
/usr/lib/tmpfiles.d/edb-uwsgi.conf
/etc/systemd/system/edb-uwsgi.service
/etc/edb-uwsgi/apps-available/pem.ini
/etc/nginx/conf.d/edb-pem.conf
```

次に、NGINX と uWSGI をリロードします。

```
systemctl daemon-reload
systemctl enable edb-uwsgi
systemctl start edb-uwsgi
systemctl restart nginx
```

1. Web サーバーとして Apache HTTPD を使用している場合、次のファイルもプライマリからコピーします。

```
/etc/httpd/conf.d/edb-pem.conf
/etc/httpd/conf.d/edb-ssl-pem.conf
```

PEM 10.3 以前からアップグレードする場合は、`edb-python312-mod-wsgi` パッケージをインストールします。 PEM 10.4 では、RHEL 9 に edb-Python 3.12 を導入し、システム Python 3.9 を置き換えます。このパッケージは、Apache HTTPD が正常に動作するために必要です。プライマリサーバーでは、`configure-pem-server.sh` は自動的にインストールしますが、レプリカサーバーでは手動でインストールする必要があります。

```
dnf install edb-python312-mod-wsgi
```

Apache HTTPD を再起動します。

```
systemctl restart httpd
```

アップグレードが完了しました。

#### 別の PEM Web アプリケーションとバックエンドデータベースサーバーを使用した展開のアップグレード

#### プライマリバックエンドのアップグレード

プライマリバックエンドデータベースサーバーで

1. PEM パッケージを更新します。

```
dnf upgrade edb-pem --allowerasing
```

1. アップグレードスクリプトを実行します。

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

#### レプリカバックエンドのアップグレード

各レプリカデータベースサーバーで

1. PEM パッケージを更新します。

```
dnf upgrade edb-pem --allowerasing
```

1. 次の構成ファイルをプライマリからコピーします。

```
/usr/edb/pem/share/.install-config
```

#### Web アプリケーションのアップグレード

各 Web アプリケーションサーバーで

1. PEM パッケージを更新します。

```
dnf upgrade edb-pem --allowerasing
```

1. アップグレードスクリプトを実行します。

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

アップグレードが完了しました。

### 6.5.2 PEM upgrade best practices and remediation

このページは、PEM アップグレードをエンドツーエンドで計画、実行、検証、および必要に応じて修復するのに役立ちます。アップグレード中にアクセス許可が失われる理由を説明し、損失をトリガーする操作をリストし、予防と修復の両方のためにすぐに実行できる SQL を提供します。

EDB は、PEM-6100 および PEM-6103 で繰り返しの障害モードを追跡します。製品側の修正は別途出荷されます。このページは、その間に顧客対応のギャップを埋め、その後もベストプラクティスのリファレンスとして役立ちます。

#### 6.5.2.1 PEM アップグレードで権限を保持する方法

アップグレード中に、PEM は、 `pem` 、`pemdata` 、および `pemhistory` スキーマのいくつかの内部ビューとファンクション-最もよく知られている `probe_target_view` 、`probe_schedule_view` 、`avail_agents` 、`avail_servers` 、およびバージョン間でシグネチャが変更される少数のファンクション。モニタリングが機能し続けるには、これらのオブジェクトは PEM ワーカーロール `pem_user` 、`pem_admin` 、および `pem_agent` で読み取り可能なままである必要があります。

PEM は PostgreSQL の ALTER DEFAULT PRIVILEGES に依存して、これらのオブジェクトのいずれかが再作成されるたびに標準の付与を自動的に適用します。PostgreSQL は、これらのルールを登録したロール—通常は元の PEM インストール多くの場合 enterprisedb または postgres を実行したロールによってキー付けられる pg_default_acl に保存します。

PEM アップグレードは、正確に 2 つの状況で許可を失う可能性があります。

- **所有者の変更。**PEM オブジェクトで REASSIGN OWNED BY <old> TO <new> または ALTER ... OWNER TO <new> を実行しました。REASSIGN OWNED BY は、オブジェクトの所有権を転送し、既存の ACL を保持しますが、pg_default_acl のエントリをコピーしません。これらは<old> にアタッチされたままです。 <new> によって作成される将来のオブジェクトアップグレード中に再作成されるビューなどは、付与を継承しません。
- **クロスロールアップグレード。**最初に PEM をインストールしたユーザーとは別の OS/データベースユーザーで configure-pem-server.sh を実行しています。インストール時に登録されたデフォルト特権ルールは、別のロールでは起動されません。

どちらの場合も、アップグレードは正常に完了しましたが、ワーカーログはすぐにレポートします。

```
ERROR: permission denied for view probe_schedule_view
ERROR: permission denied for view probe_target_view
ERROR: permission denied for function create_default_server_alerts
```

そして、すべての監視対象サーバーは PEM Web UI に **DOWN** と表示されます。

#### アップグレードする前にクラスターを確認する

PEM バックエンドデータベースで PostgreSQL スーパーユーザーとして実行します。

```
SELECT DISTINCT defaclrole::regrole AS defaults_bound_to,
       defaclnamespace::regnamespace AS schema
FROM   pg_default_acl
WHERE  defaclnamespace::regnamespace::text IN (pem,pemdata,pemhistory);
```

返されたロールが configure-pem-server.sh を実行するロールと一致する場合、アップグレードはそのままで安全に実行できます。そうでない場合は、最初に アップグレード前の準備 を処理します。

### 6.5.2.2 してはいけないこと

#### アップグレードの前

**Do**

- バックアップを取ります pg_dumpall --globals-only および pg_dump -Fc -d pem 。
- 最初に PEM をインストールした OS ユーザーと PostgreSQL ロールをメモします。そのロールとしてアップグレードを実行します。
- 上記の検証クエリを実行します。
- 可能であれば、最初に非実稼働環境でアップグレードをテストします。

しないでください

- 最初に アップグレード前の準備 を完了することなく、pem、pemdata、または pemhistory スキーマまたはそのオブジェクトの所有権を変更しないでください。これには、REASSIGN OWNED BY、ALTER ... OWNER TO、および ALTER ROLE ... RENAME TO が含まれます。
- **回避策として ``PUBLIC`` に特権を付与しないでください。すべてのデータベースユーザーに PEM の内部を公開します。pem_user 、pem_admin 、および pem_agent にのみ付与します。
- **PEM 管理のテーブル、ビュー、ファンクション、またはトリガーを直接変更しないでください。

#### アップグレード中

**Do**

- デフォルトの権限が登録されているロールとして configure-pem-server.sh を実行します。
- スクリプト出力を監視します。続行する前に停止してエラーを調査します。

しないでください

- **PEM をインストールしたユーザーとは別の OS またはデータベースユーザー**でアップグレードを実行しないでください新しいロールの アップグレード前の準備 を完了していない場合。
- アップグレード中に PEM バックエンドでスクリプトを中断したり、他の DDL を実行したりしないでください。

#### アップグレード後

**Do**

- アップグレード後の検証 で検証を実行します。
- サービスを開始してから 2 分間/var/log/pem/worker.log を監視します。
- すべての監視対象サーバーが Web UI で **UP** に移行することを確認します。

しないでください

- worker.log の permission denied エラーは無視しないでください。これらは、付与が失われ、修復が必要であることを意味します。すでに影響を受けているクラスターの修復 を参照してください。
- 検証に合格するまで、**新しいエージェントまたはプローブを追加しないでください** 。

##### 6.5.2.3 アップグレード前の準備

このセクションは、アップグレード前チェックで不一致が示された場合、つまり、PEM 所有者を変更するか、インストールとは別のロールでアップグレードを実行する場合にのみ適用します。すべての SQL を PostgreSQL スーパーユーザーとして実行します。すべてのブロックはべき等です。

#### ステップ 1-PEM サービスを停止します

```
sudo systemctl stop pemagent
sudo systemctl stop pemserver
```

#### ステップ 2-所有権を譲渡する 所有権を変更する場合のみ

```
REASSIGN OWNED BY <old_owner> TO <new_owner>;
GRANT <old_owner> TO <new_owner>;   -- lets <new_owner> inherit role memberships
```

#### ステップ 3-アップグレードロールのデフォルト特権を登録する

<upgrade_role> を、configure-pem-server.sh を実行するロールに置き換えます。

```
- - Schema: pem
ALTER DEFAULT PRIVILEGES FOR ROLE <upgrade_role> IN SCHEMA pem GRANT SELECT  ON␣
↪TABLES    TO pem_user;
ALTER DEFAULT PRIVILEGES FOR ROLE <upgrade_role> IN SCHEMA pem GRANT ALL     ON␣
↪TABLES    TO pem_admin;
ALTER DEFAULT PRIVILEGES FOR ROLE <upgrade_role> IN SCHEMA pem GRANT SELECT  ON␣
↪TABLES    TO pem_agent;
ALTER DEFAULT PRIVILEGES FOR ROLE <upgrade_role> IN SCHEMA pem GRANT USAGE   ON␣
↪SEQUENCES TO pem_user;
ALTER DEFAULT PRIVILEGES FOR ROLE <upgrade_role> IN SCHEMA pem GRANT ALL     ON␣
↪SEQUENCES TO pem_admin;
ALTER DEFAULT PRIVILEGES FOR ROLE <upgrade_role> IN SCHEMA pem GRANT EXECUTE ON␣
↪FUNCTIONS TO pem_user;
ALTER DEFAULT PRIVILEGES FOR ROLE <upgrade_role> IN SCHEMA pem GRANT ALL     ON␣
↪FUNCTIONS TO pem_admin;
ALTER DEFAULT PRIVILEGES FOR ROLE <upgrade_role> IN SCHEMA pem GRANT EXECUTE ON␣
↪FUNCTIONS TO pem_agent;

- - Schema: pemdata
ALTER DEFAULT PRIVILEGES FOR ROLE <upgrade_role> IN SCHEMA pemdata GRANT SELECT
                  ON TABLES    TO pem_user;
ALTER DEFAULT PRIVILEGES FOR ROLE <upgrade_role> IN SCHEMA pemdata GRANT ALL
                  ON TABLES    TO pem_admin;
ALTER DEFAULT PRIVILEGES FOR ROLE <upgrade_role> IN SCHEMA pemdata GRANT SELECT,␣
↪INSERT, UPDATE, DELETE ON TABLES    TO pem_agent;
ALTER DEFAULT PRIVILEGES FOR ROLE <upgrade_role> IN SCHEMA pemdata GRANT ALL
                  ON SEQUENCES TO pem_admin;
ALTER DEFAULT PRIVILEGES FOR ROLE <upgrade_role> IN SCHEMA pemdata GRANT ALL
                  ON FUNCTIONS TO pem_admin;
ALTER DEFAULT PRIVILEGES FOR ROLE <upgrade_role> IN SCHEMA pemdata GRANT EXECUTE
                  ON FUNCTIONS TO pem_agent;
```

(次のページに続く)

(前のページからの続き)

```
- - Schema: pemhistory
ALTER DEFAULT PRIVILEGES FOR ROLE <upgrade_role> IN SCHEMA pemhistory GRANT SELECT
                    ON TABLES    TO pem_user;
ALTER DEFAULT PRIVILEGES FOR ROLE <upgrade_role> IN SCHEMA pemhistory GRANT ALL
                    ON TABLES    TO pem_admin;
ALTER DEFAULT PRIVILEGES FOR ROLE <upgrade_role> IN SCHEMA pemhistory GRANT SELECT,␣
↪INSERT, UPDATE, DELETE ON TABLES    TO pem_agent;
ALTER DEFAULT PRIVILEGES FOR ROLE <upgrade_role> IN SCHEMA pemhistory GRANT ALL
                    ON SEQUENCES TO pem_admin;
ALTER DEFAULT PRIVILEGES FOR ROLE <upgrade_role> IN SCHEMA pemhistory GRANT ALL
                    ON FUNCTIONS TO pem_admin;
ALTER DEFAULT PRIVILEGES FOR ROLE <upgrade_role> IN SCHEMA pemhistory GRANT EXECUTE
                    ON FUNCTIONS TO pem_agent;
```

#### ステップ 4-孤立したデフォルトを古いロールから取り消しますオプショナル、所有権を変更した場合のみ

```
ALTER DEFAULT PRIVILEGES FOR ROLE <old_owner> IN SCHEMA pem        REVOKE ALL ON␣
↪TABLES, SEQUENCES, FUNCTIONS FROM pem_user, pem_admin, pem_agent;
ALTER DEFAULT PRIVILEGES FOR ROLE <old_owner> IN SCHEMA pemdata    REVOKE ALL ON␣
↪TABLES, SEQUENCES, FUNCTIONS FROM pem_user, pem_admin, pem_agent;
ALTER DEFAULT PRIVILEGES FOR ROLE <old_owner> IN SCHEMA pemhistory REVOKE ALL ON␣
↪TABLES, SEQUENCES, FUNCTIONS FROM pem_user, pem_admin, pem_agent;
```

#### ステップ 5-スキーマ USAGE セーフティネット

```
GRANT USAGE ON SCHEMA pem, pemdata, pemhistory TO pem_user, pem_admin, pem_agent;
```

#### ステップ 6-検証クエリとアップグレードを再実行します

アップグレードする前にクラスターを確認するからクエリーを再実行します。デフォルトが<upgrade_role>にバインドされたことを確認して、アップグレードを実行します。

```
sudo /usr/bin/dnf upgrade -y ./edb-pem*.rpm --allowerasing
sudo /usr/edb/pem/bin/configure-pem-server.sh
```

### 6.5.2.4 アップグレード後の検証

PEM バックエンドデータベースで PostgreSQL スーパーユーザーとして次を実行します。

```
- - Confirm grants on the views most commonly affected.
- - Each output should show entries for pem_user, pem_admin, and pem_agent.
\dp pem.probe_target_view
\dp pem.probe_schedule_view
\dp pem.avail_agents
```

(次のページに続く)

(前のページからの続き)

```
\dp pem.avail_servers

-- Smoke-test as pem_agent.
SET ROLE pem_agent;
SELECT count(*) FROM pem.probe_schedule_view;
SELECT count(*) FROM pem.probe_target_view;
RESET ROLE;

-- Sweep for any object pem_user cant read.
-- Zero rows = clean. Repeat with pem_admin and pem_agent.
SELECT c.relkind, n.nspname || . || c.relname AS object
FROM   pg_class c JOIN pg_namespace n ON n.oid = c.relnamespace
WHERE  n.nspname IN (pem,pemdata,pemhistory)
  AND  c.relkind IN (r,v,m)
  AND  NOT has_table_privilege(pem_user, c.oid, SELECT)
ORDER BY 1,2;
```

サービスを開始し、ログを 2 分間監視します。

```
sudo systemctl start pemserver
sudo systemctl start pemagent
tail -f /var/log/pem/worker.log
```

permission denied メッセージは表示されません。 PEM Web UI で、すべての監視対象サーバーが **UP** に移行することを確認します。

#### 6.5.2.5 すでに影響を受けているクラスターの修復

アップグレードが既に完了し、probe_target_view 、probe_schedule_view 、avail_agents 、avail_servers 、または create_default_server_alerts の permission denied エラーが表示されている場合は、PostgreSQL のスーパーユーザーとして以下のインプレース修復を実行します。正常なクラスターではべき等で安全です。

```
GRANT USAGE ON SCHEMA pem, pemdata, pemhistory TO pem_user, pem_admin, pem_agent;

-- Schema: pem
GRANT SELECT  ON ALL TABLES    IN SCHEMA pem TO pem_user, pem_agent;
GRANT ALL     ON ALL TABLES    IN SCHEMA pem TO pem_admin;
GRANT EXECUTE ON ALL FUNCTIONS IN SCHEMA pem TO pem_user, pem_admin, pem_agent;
GRANT USAGE   ON ALL SEQUENCES IN SCHEMA pem TO pem_user;
GRANT ALL     ON ALL SEQUENCES IN SCHEMA pem TO pem_admin;

-- Schema: pemdata
GRANT SELECT                         ON ALL TABLES    IN SCHEMA pemdata TO pem_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON ALL TABLES    IN SCHEMA pemdata TO pem_agent;
```

(次のページに続く)

(前のページからの続き)

```
GRANT ALL                              ON ALL TABLES    IN SCHEMA pemdata TO pem_admin;
GRANT EXECUTE                          ON ALL FUNCTIONS IN SCHEMA pemdata TO pem_user,␣
↪pem_admin, pem_agent;
GRANT ALL                              ON ALL SEQUENCES IN SCHEMA pemdata TO pem_admin;

- - Schema: pemhistory
GRANT SELECT                           ON ALL TABLES    IN SCHEMA pemhistory TO pem_
↪user;
GRANT SELECT, INSERT, UPDATE, DELETE ON ALL TABLES    IN SCHEMA pemhistory TO pem_
↪agent;
GRANT ALL                              ON ALL TABLES    IN SCHEMA pemhistory TO pem_
↪admin;
GRANT EXECUTE                          ON ALL FUNCTIONS IN SCHEMA pemhistory TO pem_
↪user, pem_admin, pem_agent;
GRANT ALL                              ON ALL SEQUENCES IN SCHEMA pemhistory TO pem_
↪admin;
```

再起動は必要ありません。次のプローブサイクル約 60 秒以内に、新しい権限が取得されます。次回のアップグレードで同じ問題が発生するのを防ぐには、そのアップグレードを実行するロールの アップグレード前の準備 も完了します。

#### 6.5.2.6 例 所有権を変更したアップグレード

これは、最も一般的な非標準のシナリオの具体的なチュートリアルです。PEM は、old_owner が所有するソースサーバーにインストールされました。クラスターは、企業標準の所有者が new_owner であるターゲットサーバーで復元されています。目標は、最終的な所有者が new_owner で PEM をアップグレードすることです。

重要なのは順序です。次の手順をこの順序で実行します。

1. バックアップを取ります してはいけないこと を参照してください。
2. サービスを停止します アップグレード前の準備 のステップ 1。
3. 所有権を転送します REASSIGN OWNED BY old_owner TO new_owner; GRANT old_owner TO new_owner; ステップ 2。
4. デフォルト FOR ROLE new_owner を登録しますステップ 3。
5. 孤立したデフォルト FOR ROLE old_owner を取り消しますステップ 4。
6. スキーマ USAGE セーフティネットを適用しますステップ 5。
7. RPM アップグレードと configure-pem-server.sh を new_owner として実行します ステップ 6。
8. アップグレード後の検証を実行します。
9. サービスを開始します。

ステップ 4 がスキップされる場合、ステップ 7 は pem.probe_target_view および pem.probe_schedule_view を new_owner として再作成します。 PostgreSQL は、 pg_default_acl で

new_owner のデフォルトを見つけず、これらはまだ old_owner にバインドされており、ビューは許可なしで思いつきます。ワーカーログには、permission denied for view ... と すでに影響を受けているクラスターの修復 のリカバリーが必要であることがすぐに示されます。

### 6.5.3 Upgrading the PEM backend Postgres database

PEM コンポーネントと PEM バックエンドデータベースを更新する場合は、バックエンドデータベースを更新する前にサーバーとエージェントで PEM コンポーネントの更新を実行します。PEM コンポーネントソフトウェアの更新の詳細については、*Upgrading a PEM installation* を参照してください。

**注釈**

- バックエンドデータベースサーバーがバージョン 13 より前の場合、最初にバックエンドデータベースサーバーをアップグレードし、次に PEM コンポーネントをアップグレードします。

- バックエンドデータベースサーバーをアップグレードした後、PEM Web インターフェイスでサーバーを作成中にこのエラーが発生した場合

```
Error - User does not have enough permission to add new server.
        Please contact the administrator to grant pem_database_server_registration␣
↪role
        to the enterprisedb user.
```

ロールを更新し、適切な権限を付与することにより、エラーを解決します。

```
UPDATE pem.roles SET rolid = pr.oid FROM pg_roles pr WHERE pr.rolname = pem_ ||␣
↪component;
```

更新プロセスでは、pg_upgrade ユーティリティを使用して、バックエンドサーバーのあるバージョンからより新しいバージョンに移行します。pg_upgrade は、サポートされている Postgres のバージョンと、同じプラットフォームでサポートされている Postgres の後続のリリース間の移行を有効にします。

pg_upgrade は、同じタイプのサーバー間のデータの転送をサポートしています。たとえば、pg_upgrade を使用して、PostgreSQL 13 バックエンドデータベースから PostgreSQL 14 バックエンドデータベースにデータを移動できますが、EDB Postgres Advanced Server 14 バックエンドデータベースには移動できません。別の種類のバックエンドデータベースに移行する場合 PostgreSQL サーバーから EDB Postgres Advanced Server などに移行する場合、*Moving the PEM server* を参照してください。

pg_upgrade の使用の詳細については、 pg_upgrade を参照してください。

1. 更新されたインストーラーをダウンロードして起動します。PostgreSQL および EDB Postgres Advanced Server のインストーラーは、EDB website から入手できます。

アップグレードするサーバーバージョンのインストーラーをダウンロードした後、PEM サーバーのホストでインストーラーを起動します。インストールウィザードの画面上の指示に従って、Postgres サーバーを構成およびインストールします。

オプションで、カスタム構築の PostgreSQL サーバーを PEM バックエンドデータベースのホストとして使用できます。ポート 5432 でリッスンする PostgreSQL バックエンドデータベースからアップグレードする場合、新しいサーバーは別のポートでリッスンするように構成する必要があります。

2. 新しいサーバーで SSL ユーティリティを構成します。新しいバックエンドデータベースは、現在のバックエンドデータベースが実行しているものと同じバージョンの sslutils を実行している必要があります。

sslutils の構成 sslutils はプラットフォーム固有です。sslutils のインストール方法の詳細については、installation documentation for your platform を参照してください。

3. 古いバックエンドデータベースと新しいバックエンドデータベースの両方のサービスを停止します。

RHEL 8.x で、コマンドラインを開き、スーパーユーザーの ID を引き継ぎます。次のコマンドを入力します。

```
systemctl <service_name> stop
```

<service_name> は、Postgres サービスの名前を指定します。

Windows では、[ Services ] ダイアログボックスを使用して Services を制御できます。サービスを停止するには 1. Control Panel で、 **System and Security > Administrative Tools** を選択します。 1. **Services** をダブルクリックします。 1. [ Services ] ダイアログボックスで、サービス名を選択し、[ **Stop** ] を選択します。

4. pg_upgrade ユーティリティを使用して、古いバックエンドデータベースと新しいバックエンドデータベース間で既存のデータのインプレース転送を実行します。サーバーが md5 認証を強制するように構成されている場合、データベースのスーパーユーザーの接続プロパティおよびパスワードを指定するエントリーを.pgpass ファイルに追加する必要があります。または、 pg_upgrade を呼び出す前に、pg_hba.conf ファイルを変更して信頼接続を許可する必要がある場合があります。.pgpass ファイルへのエントリーの作成の詳細については、PostgreSQL core documentation を参照してください。

アップグレードプロセス中に、pg_upgrade は一連のログファイルを書き込みます。クラスタ所有者は、書き込み権限があるディレクトリから pg_upgrade を呼び出す必要があります。アップグレードが正常に完了した場合、pg_upgrade はアップグレードの完了時にログファイルを削除します。pg_upgrade でアップグレードログファイルを削除したくない場合は、pg_upgrade を呼び出すときに--retain キーワードを含みます。

pg_upgrade を呼び出すには、クラスター所有者の ID を引き継ぎ、クラスター所有者が書き込み権限があるディレクトリに移動し、コマンドを実行します。

```
<path_to_pg_upgrade> pg_upgrade

-d <old_data_dir_path>

-D <new_data_dir_path>

-b <old_bin_dir_path> -B <new_bin_dir_path>

-p <old_port> -P <new_port>

-u <user_name>
```

そこで

- path_to_pg_upgrade は、pg_upgrade ユーティリティの場所を指定します。デフォルトでは、pg_upgrade は Postgres ディレクトリの下の bin ディレクトリにインストールされます。 -old_data_dir_path は、古いバックエンドデータベースのデータディレクトリへの完全なパスを

指定します。- new_data_dir_path は、新しいバックエンドデータベースのデータディレクトリへの完全なパスを指定します。- old_bin_dir_path は、古いバックエンドデータベースの bin ディレクトリへの完全なパスを指定します。- new_bin_dir_path は、古いバックエンドデータベースの bin ディレクトリへの完全なパスを指定します。- old_port は、古いサーバーがリッスンしているポートを指定します。- new_port は、新しいサーバーがリッスンするポートを指定します。 - user_name は、クラスター所有者の名前を指定します。

たとえば、次のコマンドは、 Windows システムで PostgreSQL 9.6 から PostgreSQL 11 に PEM データベースを移行するように pg_upgrade に指示しますバックエンドデータベースがデフォルトの場所にインストールされている場合。

```
C:\>"C:\Program Files\PostgreSQL1\bin\pg_upgrade.exe"

-d "C:\Program Files\PostgreSQL0\data"

-D "C:\Program Files\PostgreSQL1\data"

-b "C:\Program Files\PostgreSQL0\bin"

-B "C:\Program Files\PostgreSQL1\bin"

-p 5432 -P 5433

-U postgres
```

pg_upgrade は、呼び出されると、データを新しいバックエンドデータベースに移動する前に一貫性チェックを実行します。アップグレードが完了すると、pg_upgrade はアップグレードが完了したことを通知します。

pg_upgrade オプションの使用またはアップグレードプロセスのトラブルシューティングの詳細については、pg_upgrade を参照してください。

5. 次の証明書ファイルを、古いバックエンドデータベースの data ディレクトリから新しいバックエンドデータベースの data ディレクトリにコピーします。

ca_certificate.crt

ca_key.key

root.crt

root.crl

server.key

server.crt

ターゲットサーバーに配置されたら、ファイルに次のプラットフォーム固有の権限があることを確認します。

**Linux での権限と所有権**

| File name | Owner | Permissions |
|---|---|---|
| ca_certificate.crt | postgres | -rw------- |
| ca_key.key | postgres | -rw------- |
| root.crt | postgres | -rw------- |
| root.crl | postgres | -rw------- |
| server.key | postgres | -rw------- |
| server.crt | postgres | -rw-r--r-- |

Linux では、証明書ファイルは postgres が所有する必要があります。コマンドラインで次のコマンドを使用して、ファイルの所有権を変更できます。

```
chown postgres <file_name>
```

`file_name` は、証明書ファイルの名前を指定します。

`server.crt` ファイルの所有者のみが変更できますが、すべてのユーザーが読み取ることができます。次のコマンドを使用して、`server.crt` ファイルのファイル許可を設定できます。

```
chmod 644 server.crt
```

他の証明書ファイルの所有者のみがファイルを変更または読み取ることができます。次のコマンドを使用して、ファイルのアクセス許可を設定できます。

```
chmod 600 <file_name>
```

`file_name` は、ファイルの名前を指定します。

**Windows での権限と所有権**

Windows では、ターゲットホストで PEM サーバーとバックエンドデータベースのインストールを実行したサービスアカウントは、ソースホストから移動した証明書ファイルを所有している必要があります。インストーラーのコンテキストメニューから **Run as Administrator** を使用して PEM サーバーと Postgres インストーラーを呼び出した場合、証明書ファイルの所有者は Administrators です。

Windows でファイルのアクセス許可を確認および変更するには、ファイル名を右クリックし、**プロパティ** を選択します。

[ **セキュリティ** ] タブで、グループまたはユーザー名を選択して、割り当てられたアクセス許可を表示します。**Edit** または **Advanced** を選択して、選択したユーザーに関連付けられている権限を変更できるダイアログボックスを開きます。

6. `postgresql.conf` ファイルには、サーバーの動作を指定するパラメーター設定が含まれています。古いサーバーの `postgresql.conf` ファイルで指定された構成と一致するように、新しいサーバーの `postgresql.conf` ファイルを変更します。

選択したエディターを使用して、新しいサーバーの `postgresql.conf` ファイルを更新します。次のパラメーターを変更します。

- 元のバックエンドデータベースによって監視されるポートでリッスンする `port` パラメーター通常は `5432` に設定します。

- on に設定する ssl パラメーター

また、次のパラメーターが有効になっていることを確認する必要があります。パラメーターがコメントアウトされている場合は、各 postgresql.conf ファイルエントリーの前にあるポンド記号を削除します。

- ssl_cert_file = 'server.crt' # (change requires restart)
- ssl_key_file = 'server.key' # (change requires restart)
- ssl_ca_file = 'root.crt' # (change requires restart)
- ssl_crl_file = 'root.crl'

インストールには、新しいバックエンドデータベースが古いバックエンドデータベースのように動作するように変更が必要な他のパラメーター設定がある場合があります。postgresql.conf ファイルを注意深く確認して、新しいサーバーの構成が古いサーバーの構成と一致することを確認します。

7. pg_hba.conf ファイルには、サーバーがホストベースの認証を強制する方法を指定するパラメーター設定が含まれています。PEM サーバーをインストールすると、インストーラーは pg_hba.conf ファイルを変更し、ファイルの先頭にエントリを追加します。

### 6.5.4 Adding entries for PEM Agens and admins to connect to PEM server

```
hostssl pem +pem_user 192.168.2.0/24 md5

hostssl pem +pem_agent 192.168.2.0/24 cert
```

### 6.5.5 Adding entries (localhost) for PEM Agens and admins to connect to PEM server

```
hostssl pem +pem_user 127.0.0.1/32 md5

hostssl postgres +pem_user 127.0.0.1/32 md5

hostssl pem +pem_user 127.0.0.1/32 md5

hostssl pem +pem_agent 127.0.0.1/32 cert
```

選択したエディターを使用して、古いサーバーの pg_hba.conf ファイルから新しいサーバーの pg_hba.conf ファイルにエントリをコピーします。

8. 新しいバックエンドデータベースのサービスを再起動します。

RHEL 8.x では、スーパーユーザーとしてコマンドラインで次のように入力します。

```
systemctl restart <service_name>
```

service_name は、バックエンドデータベースサーバーの名前です。

Windows を使用している場合は、 **Services** ダイアログボックスを使用してサービスを制御できます。1. Control Panel で、**System and Security > Administrative Tools** を選択します。 1. **Services** アイコンをダブルクリックします。 1. [ Services ] ダイアログボックスで、サービス名を選択し、サービスを開始します。

### 6.5.6 Upgrading SQL Profiler

プラグインに含まれる SQL ファイルをロードしてファンクションが作成される SQL プロファイル v4.0 または以前のバージョンからアップグレードする場合、最初に拡張機能に移行し、後で 4.1 に更新します。

```
#  running as superuser

CREATE EXTENSION sql_profiler VERSION 4.0;

ALTER EXTENSION sql_profiler UPDATE TO 4.1;
```

### 6.5.7 Upgrading the PEM web server

PEM Web サーバーは、PEM Web アプリケーションを提供するために使用されます。Linux では、Web サーバーは NGINX または Apache HTTPD であり、PEM 10.0 以降の新規インストールのデフォルトは NGINX です。PEM は Web サーバーをバンドルせず、代わりに、選択した Web サーバーのシステムパッケージマネージャーが提供するパッケージを使用します。必要に応じて、パッケージマネージャーを使用して *Linux での Web サーバーのアップグレード* できます。

Windows では、PEM インストーラーは、PEM-HTTPD と呼ばれる Apache HTTPD のバージョンをバンドルしています。したがって、EDB インストーラーを使用してのみ *update the web server* できます。

#### 6.5.7.1 Linux での Web サーバーのアップグレード

**重要**

Red Hat Enterprise LinuxRHEL、Rocky Linux、AlmaLinux、または Oracle Linux を使用しており、Web サーバーを更新して脆弱性に対処する場合、 *RHEL および RHEL 派生製品のバージョン管理* を読んでください。

Linux を実行している場合は、サーバー上の他のシステムソフトウェアとともにウェブサーバーを管理します。パッケージマネージャーから入手可能な最新バージョンを常に使用することをお勧めします。

アップグレードする前に PEM を停止したり、ファイルをバックアップしたりする必要はありません。パッケージマネージャーからアップグレードを開始します。

##### RHEL / Rocky Linux / AlmaLinux / Oracle Linux

次のコマンドを実行し、必要に応じて `nginx` または `httpd` を`<web-server>` に置き換えます。

```
sudo yum update <web-server>
```

##### Debian / Ubuntu

次のコマンドを実行し、必要に応じて `nginx` または `apache2` を`<web-server>` に置き換えます。

```
sudo apt update
sudo apt upgrade <web-server>
```

#### SLES

次のコマンドを実行し、必要に応じて nginx または apache2 を<web-server> に置き換えます。

```
sudo zypper update <web-server>
```

#### RHEL および RHEL 派生製品のバージョン管理

RHEL またはその他の RHEL 派生から入手可能な NGINX または HTTPD の最新バージョンのバージョン番号が、最新のコミュニティリリースよりもかなり古いことに気づくでしょう。具体的には、RHEL 8 は NGINX 1.22 および HTTPD 2.4.37 を提供します。 RHEL 9 は、NGINX 1.22 および HTTPD 2.4.62 を提供します。 Red Hat は、RHEL 用のカスタム NGINX および HTTPD パッケージをビルドし、セキュリティ修正をバックポートしていることを理解することが非常に重要です。

このため、コミュニティリリースに存在する脆弱性が同じバージョンの RHEL パッケージに存在するとは考えないでください。この理由で HTTPD の新しいバージョンをインストールしようとしている場合、それはほぼ確実に必要ありません。詳細については、Red Hat ナレッジベースの NGINX versions supported by Red Hat および Apache HTTPD versions supported by Red Hat 記事を参照してください。

> **警告**
>
> Red Hat は、コミュニティからの NGINX または HTTPD パッケージ、またはソースからビルドをサポートしていません。 RHEL システムで NGINX または HTTPD の他のソースを使用することを検討している場合は、 Apache HTTPD versions supported by Red Hat 記事を読んで、その影響を理解してください。

### 6.5.7.2 Windows での PEM-HTTPD のアップグレード

PEM-HTTPD は PEM Windows インストーラーにバンドルされており、Stack Builder から入手できます。一般に、PEM インストールを定期的に更新することをお勧めします。PEM アプリケーションを更新すると、PEM-HTTPD インストールが自動的に更新されます。

PEM を更新したくない場合、または PEM インストーラーではまだ利用できない PEM-HTTPD リリースを適用したくない場合は、StackBuilder を使用できます。

StackBuilder を使用してアップグレードするには、Windows 用の最新の PostgreSQL または EDB Advanced Server インストーラーをダウンロードし、 Stack Builder を使用します

または StackBuilder Plus を使用して PEM-HTTPD を更新します。 PEM-HTTPD は、Web Development カテゴリの下にあります。

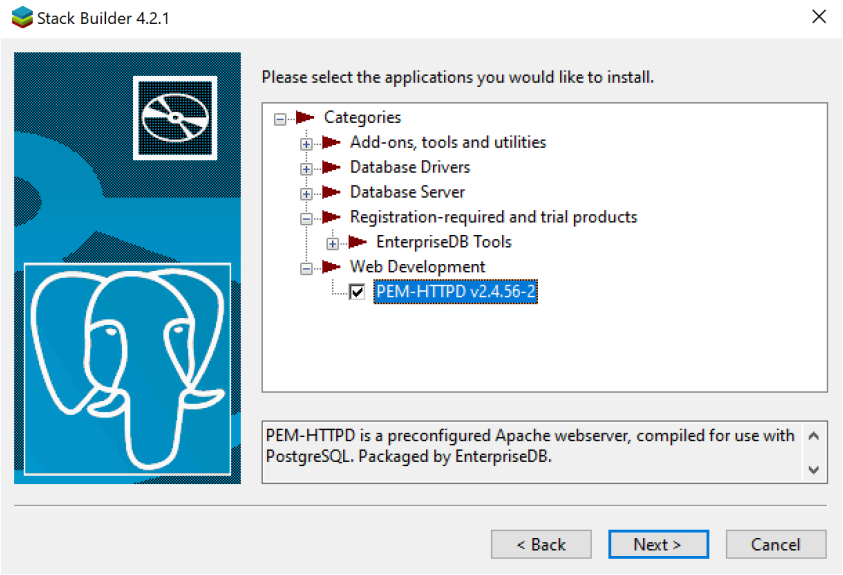



図 5: PEM-HTTPD in Stack Builder

### 6.5.8 Moving the PEM server

PEM サーバーをあるホストマシンから新しいホストマシンに移動できます。新しいホストターゲット上の PEM サーバーは、元のホストソースと同じバージョンの PEM サーバーインストーラーでインストールする必要があります。同じインストーラーバージョンを使用しない場合、スキーマ不一致エラーが発生する場合があります。

ターゲットサーバーのバックエンドデータベース PostgreSQL または EDB Postgres Advanced Server は、ソース PEM サーバーのバックエンドデータベースと同じタイプとバージョン、または異なるタイプとバージョンを持つことができます。PostgreSQL ホスト上にある PEM サーバーは、 EDB Postgres Advanced Server ホストに移行できます。その逆も可能です。

サーバーの移行を開始する前に、ソースホスト、ターゲットホスト、および PEM エージェントのホスト間のファイアウォールがサービス間の接続を許可していることを確認してください。

1. ターゲットホストを準備します。

ターゲットホストで PEM サーバーのインストーラーを起動します。ソース PEM サーバーのインストール時に使用したものと同じバージョンの PEM サーバーインストーラーを使用する必要があります。

ターゲットサーバーのバックエンドデータベースは、ソースのバックエンドデータベースとは異なるバージョンまたはタイプを持つことができます。新しい PEM サーバーが元のサーバーと同じタイプのバックエンドデータベースにない場合、新しいバージョンの sslutils 拡張機能が新しいサーバーホストにインストールされていることを確認する必要があります。PEM インストーラーと共に配布される sslutils のバージョンは、 EDB website から無料でダウンロードできます。

PEM サーバーまたは sslutils 拡張機能のインストールについては、PEM installation steps を参照してください。

2. 新しい PEM サーバーから既存のスキーマを削除します。

移行プロセスは、ターゲット PEM サーバー上のソース PEM サーバーから `pem`、`pemdata`、および `pemhistory` スキーマを再作成します。移動の準備をするには、psql クライアントを使用して、ターゲットホスト上の `pem`

データベースからこれらのスキーマを削除します。コマンドラインで、または **Postgres Enterprise Manager > SQL Shell (psql)** を選択して、psql クライアントを開くことができます。

psql クライアントが開いたら、データベーススーパーユーザーとして pem バックエンドデータベースに接続します。ターゲットホスト上の pem データベースに接続した後、スキーマを削除します。

```
DROP SCHEMA pem CASCADE;

DROP SCHEMA pemdata CASCADE;

DROP SCHEMA pemhistory CASCADE;
```

スキーマを削除するときは、CASCADE キーワードを含め、すべての依存オブジェクトを削除するようにサーバーに指示する必要があります。コマンドを実行すると、psql クライアントは依存オブジェクトのリストを表示します。クライアントは、DROP SCHEMA を表示することにより、各スキーマが削除されることを確認します。

3. 新しい PEM サーバーで PEM エージェントを準備します。

PEM サーバーを移動する前に、ソース PEM サーバーによって監視されているエージェントの数を特定し、ターゲットサーバーでその数のエージェント 1 つ未満の ID を作成する必要があります。PEM サーバーによって監視される PEM エージェントの合計数を検出するには、psql クライアントを使用してソースホスト上の pem データベースに接続し、pem.agent テーブルを照会します。

```
SELECT id FROM pem.agent WHERE active = true;
```

元の PEM サーバーにあるエージェントの数から 1 つ未満を手動で作成する必要があります。PEM サーバーインストーラーは、ターゲットホストに 1 つのエージェントを作成します。たとえば、ソースサーバーに 3 つのエージェントが含まれている場合、さらに 2 つのエージェントを手動で作成する必要があります。ターゲットサーバー上の pem データベースとの psql セッションを開き、必要なエージェントを作成します。

```
CREATE USER agent<X>;
```

<X> はエージェント番号を指定します。 agent1 は、PEM サーバーインストーラーによってターゲットホストに作成されます。

次に、 GRANT コマンドを使用して、ターゲット PEM サーバーにある各エージェントに pem_agent 権限を割り当てます。

```
GRANT pem_agent TO agent<X>;
```

<X> はエージェント番号を指定します。

4. ソース PEM サーバーのバックアップスクリプトを生成します。

pg_dump ユーティリティを使用して、ターゲットホストで pem データベースを再作成するために必要なコマンドを含むスクリプトを生成できます。デフォルトでは、pg_dump は Postgres インストールの下の bin ディレクトリにインストールされます。 pg_dump を呼び出すには、bin ディレクトリで次のように入力します。

```
pg_dump -U <user_name> <db_name> > <file_name>
```

そこで

- <user_name> は、PEM バックエンドデータベースのデータベーススーパーユーザーの名前を指定します。- <db_name> は、PEM バックエンドデータベースの名前を指定します。 - <file_name> は、pg_dump によって生成されたスクリプトの名前を指定します。

プロンプトが表示されたら、指定したユーザーに関連付けられたパスワードを入力します。

表示されているコマンドは、 pg_dump に実行時に pem データベースを再作成するスクリプトを生成するように指示します。スクリプトの名前は backup.sql で、 tmp ディレクトリに作成されます。pg_dump は、ユーザー postgres の資格情報を使用してサーバーに接続しています。

pg_dump ユーティリティを呼び出しても、現在のデータベースユーザーは中断されません。

5. バックアップをターゲットホストに移動します。

pg_dump ユーティリティによって生成されたスクリプトを PEM サーバーのターゲットホストに移動します。

6. ターゲットホストでバックアップを復元します。

ターゲットホストで、Postgres バックエンドデータベースインストールディレクトリの下の bin ディレクトリで、psql 起動し、pg_dump ユーティリティによって生成されたスクリプトを実行します。

```
psql -U <user_name> -d pem -f <file_name>
```

そこで

- <user_name> は、データベースのスーパーユーザーの名前を指定します。指定されたユーザーは、バックエンドデータベースに対する接続特権を持っている必要があります。- <file_name> は、pg_dump によって生成されたバックアップスクリプトへの完全なパスを指定します。

プロンプトが表示されたら、データベースのスーパーユーザーに関連付けられたパスワードを入力します。

この例では、 psql クライアントを使用して backup.sql という名前のスクリプトを呼び出し、 pem データベースを再作成します。スクリプトは、データベーススーパーユーザー postgres に関連付けられた特権を使用して呼び出されます。

7. ターゲットホストでデータベースサーバーを停止します。

RHEL 8.x で PEM サーバーを停止するには、次のコマンドを使用します。

```
systemctl stop <service_name>
```

<service_name> は、バックエンドデータベースサーバーの名前を指定します。PostgreSQL バックエンドデータベースの場合、サービス名は postgresql-<x> です。 EDB Postgres Advanced Server バックエンドデータベースの場合、サービス名は edb-as-<X> で、<X> はバージョン番号を指定します。

Windows を使用している場合は、[ Services ] ダイアログボックスを使用して Services を制御できます。 [ Services ] ダイアログボックスを開くには、Control Panel から **System and Security > Administrative Tools** を選択します。 **Services** アイコンをダブルクリックします。 [ Services ] ダイアログボックスで、リストからサービス名を選択し、[ **Stop** ] を選択します。

8. 証明書ファイルをターゲットホストにコピーします。

ターゲットホストのインストール時に作成される証明書ファイルを、ソースホストの証明書ファイルに置き換える必要があります。次のファイルをソース PEM サーバーからターゲット PEM サーバーにコピーします。

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

ターゲットクラスタのバックエンドデータベースを提供する Postgres インストールの下の data ディレクトリにファイルをコピーします。

Linux では、ファイルは次の場所にあります。

```
/var/lib/pgsql/<X>/data/
```

Windows では、ファイルは次の場所にあります。

```
C:\Program Files\PostgreSQL\<X>\data
```

そこで

<X> は、システム上の PostgresSQL のバージョンを指定します。

ファイルは既にターゲットクラスタに存在します。コピーを実行する前に既存のファイルを削除するか、既存のファイルをソースサーバーのファイルで上書きします。ターゲットサーバーに配置されたら、ファイルには、表示されているプラットフォーム固有の権限が必要です。

**Linux の場合**

| File name | Owner | Permissions |
|---|---|---|
| ca_certificate.crt | postgres | -rw------- |
| ca_key.key | postgres | -rw------- |
| root.crt | postgres | -rw------- |
| root.crl | postgres | -rw------- |
| server.key | postgres | -rw------- |
| server.crt | postgres | -rw-r--r-- |

Linux では、証明書ファイルは postgres が所有する必要があります。次のコマンドを使用して、ファイルの所有権を変更します。

```
chown postgres <file_name>
```

file_name は、証明書ファイルの名前を指定します。

server.crt ファイルの所有者のみがファイルを変更できますが、すべてのユーザーが読み取ることができます。次のコマンドを使用して、server.crt ファイルのファイル許可を設定します。

```
chmod 644 server.crt
```

他の証明書ファイルの所有者のみがファイルを変更または読み取ることができます。次のコマンドを使用して、ファイルのアクセス許可を設定します。

```
chmod 600 <file_name>
```

file_name は、ファイルの名前を指定します。

**Windows の場合*

Windows では、ターゲットホストで PEM サーバーとバックエンドデータベースのインストールを実行したサービスアカウントは、ソースホストから移動した証明書ファイルを所有している必要があります。インストーラーコンテキストメニューから **Run as Administrator** を使用して PEM サーバーと Postgres インストーラーを呼び出した場合、証明書ファイルの所有者は Administrators です。

Windows でファイルのアクセス許可を確認および変更するには、ファイル名を右クリックし、**プロパティ** を選択します。 [ **Security** ] タブで、グループまたはユーザー名を選択して、割り当てられたアクセス許可を表示します。**Edit** または **Advanced** を選択して、選択したユーザーに関連付けられている権限を変更できるダイアログボックスを開きます。

9. PEM エージェント証明書ファイルを PEM サーバーホストに移動します。

ソース PEM サーバーの PEM エージェントが使用する証明書ファイルをターゲットホストに移動する必要があります。この手順はプラットフォーム固有です。

**Linux の場合**

agent1.key および agent1.crt ファイルをソースホストからターゲットホストにコピーします。デフォルトでは、Linux では、ファイルは/root/.pem にインストールされます。ファイルをターゲットホスト上の同じディレクトリにコピーします。

ファイルの所有権とファイルのアクセス許可を次のように設定する必要があります。

| File name | Owner | Permissions |
|---|---|---|
| agent1.key | root | -rw------- |
| agent1.crt | root | -rw-r--r-- |

必要に応じて、/root/.pem に移動し、次のコマンドを使用して agent1.key ファイルの権限と所有権を変更します。

```
chmod 600 agent1.key

chown root agent1.key
```

次のコマンドを使用して、agent1.crt ファイルの権限と所有権を変更します。

```
chmod 644 agent1.crt

chown root agent1.crt
```

**Windows の場合*

agent1.key および agent1.crt ファイルをソースホストからターゲットホストにコピーします。Windows では、ファイルは次の場所にあります。

```
C:\Users\<user_name>\AppData\Roaming\pem
```

user_name は、PEM インストーラーを呼び出したユーザーの名前です。

ターゲットマシン上の証明書ファイルに関連付けられている所有権と権限は、ソースマシン上の証明書ファイルの所有権と権限と一致する必要があります。インストーラーコンテキストメニューの **Run as Administrator** を使用して PEM サーバーと Postgres インストーラーを呼び出した場合、エージェント証明書ファイルの所有者は Administrators です。

Windows でファイルのアクセス許可を確認および変更するには、ファイル名を右クリックし、**プロパティ**を選択します。 [ **Security** ] タブで、グループまたはユーザー名を選択して、割り当てられたアクセス許可を表示します。**Edit** または **Advanced** を選択して、選択したユーザーに関連付けられている権限を変更できるダイアログボックスを開きます。

10. ターゲットホスト上の pg_hba.conf ファイルを更新します。

ターゲットホスト上の pg_hba.conf ファイルを変更して、各 PEM エージェントからの接続を許可します。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。

11. ターゲットホストでサーバーを起動します。

pg_hba.conf ファイルを変更した後、サーバーを再起動して変更を有効にする必要があります。

Linux でデータベースサーバーを再起動するには、次のコマンドを使用します。

```
/etc/init.d/<service_name> start
```

service_name は、バックエンドデータベースサーバーの名前です。

Windows では、[ Services ] ダイアログボックスを使用して Services を制御できます。 [ Services ] ダイアログボックスを開くには、Control Panel で **System and Security > Administrative Tools** を選択します。 **Services** アイコンをダブルクリックします。 [ Services ] ダイアログボックスが開いたら、リストからサービス名を選択し、サービスを開始します。

12. 監視対象エージェントを新しい PEM サーバーホストに接続します。

既存の PEM エージェントに新しい PEM サーバーホストに接続するよう指示するには、以下を行う必要があります。

- PEM エージェントホストが新しい PEM サーバーホストに接続できることを確認します。
- PEM エージェントまたはエージェント構成ファイルを使用して各 Windows ホストのレジストリを変更する PEM エージェントを使用する各 Linux ホストで、新しい PEM サーバーの IP アドレスとポートを指定します。

- PEM エージェントのサービスを再起動します。これらの手順はプラットフォーム固有です。- *PEM エージェントが Linux にある場合* - *PEM エージェントが Windows にある場合*

#### 6.5.8.1 PEM エージェントが Linux にある場合

エディターの選択を使用して、 `agent.cfg` ファイルを変更し、`pem_host` および `pem_port` パラメーターで PEM サーバーの新しい IP アドレスとポート番号を指定します。

デフォルトでは、 `agent.cfg` ファイルは次の場所にあります。

`/usr/edb/pem/agent/etc/agent.cfg`

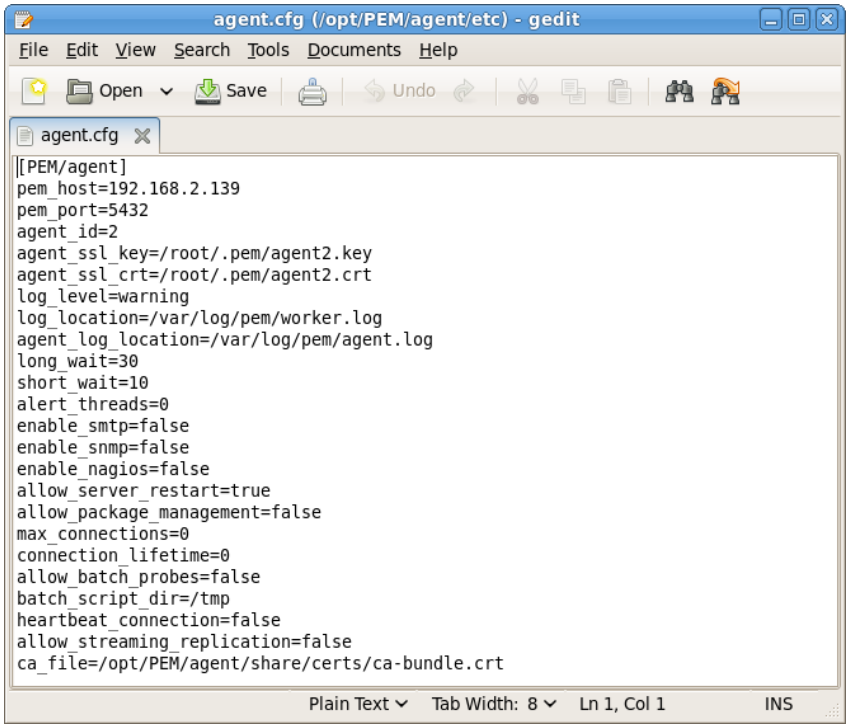



図 6: The agent.cfg file

`agent.cfg` ファイルを変更した後、PEM エージェントサービスを再起動する必要があります。Linux コマンドラインで `pemagent` サービススクリプトを使用して、サービスをリスタートできます。

```
/etc/init.d/pemagent restart
```

#### 6.5.8.2 PEM エージェントが Windows にある場合

監視対象ノードの Windows レジストリを変更する前に、PEM エージェントのホスト上のファイアウォールが PEM サーバーへの接続を許可していることを確認してください。PEM エージェントホストが PEM サーバーホストに接続できることを確認した後、Windows Registry Editor を使用して `PEM_HOST` および `PEM_PORT` エントリを確認および編集し、PEM サーバーが使用するホストおよびポートを正しく識別できることを確認できます。Registry Editor を開くには、 Windows の [ Run ] ダイアログボックスまたは Windows の [スタート] メニューの検索ボックスに `regedit` と入力します。レジストリツリーコントロールをナビゲートして、レジストリエントリを表示または変更します。

PEM エージェントのレジストリエントリは次の場所にあります。HKEY_LOCAL_MACHINE\Software\EnterpriseDB\PEM\agent

PEM_HOST および PEM_PORT エントリは、ターゲットホスト上の新しい PEM サーバーのアドレスとポート番号を指定する必要があります。レジストリエントリを変更するには、エントリ名を右クリックし、コンテキストメニューから **Modify** を選択します。次に、[ Edit String の Edit String ] ダイアログボックスを使用して、エントリの値を変更します。完了したら、[ **OK** ] を選択します。

レジストリを変更した後、PEM エージェントのサービスを再起動する必要があります。Windows コントロールパネルからアクセスする [ Services ] ダイアログボックスを使用して、Postgres Enterprise Manager - pemagent サービスを再起動できます。

サーバーを稼働した後、インストールされている PEM クライアントの接続プロパティを変更して、PEM サーバー、エージェント、監視対象サーバーの新しいホストに接続します。

> **注釈**
>
> サーバーを稼働した後、PEM Web インタフェースでサーバーを作成中にこのエラーが発生した場合

```
Error - User does not have enough permission to add new server.
        Please contact the administrator to grant pem_database_server_registration␣
↪role
        to the enterprisedb user.
```

ロールを更新し、適切な権限を付与することにより、エラーを解決します。

```
UPDATE pem.roles SET rolid = pr.oid FROM pg_roles pr WHERE pr.rolname = pem_ ||␣
↪component;
```

## 6.6 Splitting an existing PEM server

既存の PEM インストールを 2 つのインストールに分割できます。分割は、元の PEM インスタンスが過負荷になり、2 つのサーバーに負荷を分割する場合に役立ちます。データベース資産を再構築しており、監視にセグメント的なアプローチを採用したい場合にも役立ちます。

新しいインスタンスをそれぞれ分割することにより、既存の PEM インスタンスを 3 つ以上のインスタンスに分割できます。新しい PEM インスタンスを設定し、監視対象サーバーとエージェントの一部を新しいインスタンスに移動することもできます。

> **注釈**
>
> 例のコマンドは PostgreSQL 16 用です。環境に応じてサービス名とファイルパスを変更します。

### 6.6.1 新しいホストに PEM をインストールする

!!!重要 PEM と Postgres のバージョンは一致する必要があります

新しい環境の Postgres と PEM のバージョンが、既存の環境で使用されているバージョンと一致することを確認します。

2. *Configuring the PEM server on Linux* 。

### 6.6.2 既存の PEM インスタンスからすべてのデータをコピーします

新しい PEM ホストでは

1. Postgres および PEM エージェントサービスを停止します。

```
systemctl stop postgresql-16
systemctl stop pemagent
```

2. Postgres データディレクトリを削除します。

```
rm -rf /var/lib/pgsql/16/data/*
```

3. 新しいインスタンスに既存の PEM データのクローンを作成します。

```
su - postgres
pg_basebackup -h <OLD_PEM_IP> -p <OLD_PEM_SERVER_PORT> -U postgres \
-D /var/lib/pgsql/16/data -Fp -Xs -P
```

### 6.6.3 新しい PEM インスタンスの pg_hba.conf ファイルを更新します

この時点で、新しい PEM インスタンスの `pg_hba.conf` ファイルは、既存の PEM インスタンスにあるファイルのコピーです。既存の PEM サーバー `host ... <old_ip> ...` の外部 IP を参照するエントリがある場合は、必要に応じて新しい PEM サーバーの IP アドレスまたは `localhost` /`127.0.0.1` を使用するように変更します。

これで、Postgres および PEM エージェントサービスを開始できるようになりました。後で再度再起動が必要になる場合があります。

```
systemctl start postgresql-16
systemctl start pemagent
```

### 6.6.4 新しいサーバーで SSL を構成し、PEM エージェントを再構成します

SSL の構成方法は、SSL 証明書を管理する方法によって異なります。4 つのアプローチから、ニーズに合ったものを選択してください。

これらのシナリオはすべて、一意のエージェントユーザーまたは単一の共通エージェントユーザーを使用している場合に実行可能です。

!!!注 外部認証局

外部の認証局によって提供または署名された SSL 証明書を排他的に使用する場合、アプローチ *4* –外部 *CA* からの証明書のみを使用する に移動します。

これらの手順は、Postgres への接続に SSL を構成するための手順です。 PEM Web アプリケーションの場合、サーバーの構成中に新しい自己署名証明書を生成します。次の方法で Web サーバー証明書を置き換えることができます the documentation 。

両方の PEM インスタンスで同じサーバー証明書を使用する場合、このアプローチを使用します。このアプローチでは、SSL 設定またはファイルを変更する必要はありません。

!!!注

一般に、各 PEM サーバーが一意の ID を持つことを保証するため、アプローチ 1 よりもアプローチ 2 をお勧めします。これは良いプラクティスであり、`verify-full` などのより厳密なクライアント SSL モードを使用する場合に必要です。

2. 新しいシステムで Postgres と PEM エージェントサービスを再起動します。

3. 新しい PEM インスタンスに移行するすべてのエージェントについて、すべてのエージェントの `agent.cfg` ファイルの `pemhost` パラメーターを新しい PEM インスタンスのアドレスに更新します。

4. 影響を受ける PEM エージェントを再起動します。

#### 6.6.4.1 アプローチ 2 –異なるサーバー証明書を使用するが、同じ CA を使用する

2 つの PEM サーバーに異なるサーバー証明書を必要とし、両方の PEM サーバーが単一の認証局として機能できるようにする場合、このアプローチを使用します。

このアプローチを使用すると、いずれかの PEM サーバーが発行したエージェント証明書は同じように署名され、両方のサーバーによって信頼されます。これは、サーバーが明確な ID を持つことを保証しながら、エージェント証明書を置き換える必要を回避するため便利です。

1. Postgres データディレクトリにある既存の `server.crt` および `server.key` をバックアップします。

2. 新しい PEM サーバーの新しい `server.key` および CSR を生成します。

```
openssl genrsa -out server.key 4096
openssl req -new -key server.key -out server.csr -subj /C=your_country/ST=your_state/
↪L=your_locality/O=your_org/CN=common_name_for_server
```

3. CSR に署名し、既存の CA 証明書を使用してサーバー証明書を生成します。

```
openssl x509 -req -days 365 -in server.csr -CA ca_certificate.crt -CAkey ca_key.key -
↪CAcreateserial -out server.crt
```

4. 生成された `server.crt` ファイルの所有者と権限を変更します。

```
chown postgres /var/lib/pgsql/16/data/server.crt
chmod 600 /var/lib/pgsql/16/data/server.crt
```

5. 新しいシステムで Postgres および PEM エージェントサービスを再起動します。

6. 新しい PEM インスタンスに移行するすべてのエージェントについて、すべてのエージェントの `agent.cfg` ファイルの `pemhost` パラメーターを新しい PEM インスタンスのアドレスに更新します。

7. 影響を受ける PEM エージェントを再起動します。

#### 6.6.4.2 アプローチ 3 –すべてのサーバーとエージェントの証明書を置き換えます

2 つのサーバーに完全に異なるサーバー証明書と認証局を持ちたい場合に、このアプローチを使用します。このアプローチを使用すると、特定のエージェントはその「親」PEM サーバーでのみ認証できます。2 つの PEM インスタンスには、共通の SSL 構成はありません。

!!!重要

このアプローチでは、すべてのエージェントに対して新しいクライアント証明書を生成する必要があります。これを行うことができるオーケストレーションツールをお持ちでない限り、または common agent user を使用している場合を除き、これは多大な労力になる可能性があります。

2. 新しいシステムで Postgres および PEM エージェントサービスを再起動します。

3. 新しい PEM インスタンスに移行するすべてのエージェントについて、すべてのエージェントの `agent.cfg` ファイルの `pemhost` パラメーターを新しい PEM インスタンスのアドレスに更新します。

4. 各エージェントユーザーごとに *Regenerating the agent SSL certificates for PEM using sslutils* を作成し、新しい証明書とキーを適切な場所にコピーします。PEM サーバー上のエージェントの証明書とキーも再生成する必要があります。

5. 影響を受ける PEM エージェントを再起動します。

#### 6.6.4.3 アプローチ 4 –外部 CA からの証明書のみを使用する

外部の認証局によって提供または署名された SSL 証明書を排他的に使用する場合、このアプローチを使用します。

既存のエージェント証明書は、Postgres サーバー CA 証明書に含まれる信頼できる機関によって発行されたため、両方の PEM サーバーはすべてのエージェント証明書を検証できます。つまり、すべてのエージェントが既にいずれかの PEM サーバーに接続できることを意味します。新しい PEM サーバーの場合のみ、新しいサーバー証明書を要求する必要があります。

1. プロバイダーから新しいサーバー証明書とキーを取得し、新しい PEM インスタンスの詳細を提供します。

2. `server.crt` および `server.key` または現在 `ssl_server_cert` および `ssl_server_key` で指定されているファイルをそれぞれ新しい証明書とキーに置き換えます。

3. Postgres を再起動します。

### 6.6.5 各 PEM インスタンスで切断されたエージェントを無効にする

これで、2 つの PEM インスタンスができ、既存のエージェントがそれらに分割されました。ただし、両方のインスタンスには、すべてのエージェントのレコードが含まれています。各インスタンスに接続されていないエージェントを非アクティブとしてマーキングすることにより、PEM インスタンスをクリーンアップして、リソースの無駄と誤ったアラートを削減します。

PEM Web アプリケーションで、エージェントとその監視対象サーバーを削除します。Object Explorer で、各項目を右クリックし、 **Remove** を選択します。

または、 SQL を介してこのアクションを実行できます。 pem データベースで次を実行します。

```
UPDATE pem.agent SET active = false WHERE id in (list_of_ids);
UPDATE pem.server SET active = false WHERE id in (list_of_ids);
```

### 6.6.6 PEM インスタンスを検証する

新しいインスタンスの PEM モニタリングダッシュボードを開きます。

- 必要なサーバーとエージェントのみが表示されていることを確認します。
- これらのサーバー/エージェントがアクティブなステータスを示し、メトリックを送信していることを確認します。
- エージェントログで、接続エラーまたは認証エラーがないかを確認します。

## 6.7 Uninstalling

PEM コンポーネントをアンインストールするには、以下を参照してください。

- *Uninstalling Postgres Enterprise Manager components on Linux*
- *Uninstalling Postgres Enterprise Manager components on Windows*

### 6.7.1 Uninstalling Postgres Enterprise Manager components on Linux

PEM サーバーまたはエージェントのアンインストールのプロセスはプラットフォーム固有です。PEM サーバーのパッケージの名前は edb-pem-server で、PEM エージェントのパッケージの名前は edb-pem-agent です。

ホストから PEM サーバーパッケージをアンインストールしても、同じホストにインストールされている PEM エージェントパッケージはアンインストールされません。ただし、PEM エージェントパッケージをアンインストールすると、同じホストにインストールされている PEM サーバーパッケージもアンインストールされます。

> **注釈**
>
> PEM エージェントをアンインストールする前に、最初にエージェントの登録を解除する必要があります。 *pemworker* コマンドラインユーティリティを使用して、エージェントの登録を解除できます。その後、アンインストール手順を続行できます。

#### 6.7.1.1 RHEL、Rocky Linux、または AlmaLinux ホストからの PEM コンポーネントのアンインストール

rpm コマンドのバリエーションを使用して、Rocky Linux/AlmaLinux/RHEL 8.x ホストからインストールされたパッケージを削除できます。また、dnf remove コマンドを使用して、インストールされたパッケージを Rocky Linux/AlmaLinux/RHEL 8.x ホストから削除することもできます。パッケージを削除しても PEM データディレクトリは損傷しないことに注意してください。

rpm コマンドを呼び出して、インストールされたパッケージを削除するときに-e オプションを含めます。コマンド構文は次のとおりです。

```
rpm -e package_name
```

package_name は、削除するパッケージの名前です。

- dnf remove コマンドを使用して、pem サーバーまたはエージェントと、Rocky Linux または AlmaLinux または RHEL 8.x ホスト上の edb-pem および edb-pem-docs 依存関係を削除できます。パッケージを削除するには、ターミナルウィンドウを開き、スーパーユーザー権限を引き受けて、次のコマンドを入力します。

```
dnf remove package_name
```

#### 6.7.1.2 Debian または Ubuntu ホストからの PEM コンポーネントのアンインストール

apt-get remove または apt-get purge コマンドを使用して、Debian または Ubuntu ホストから PEM サーバーまたはエージェントパッケージをアンインストールできます。

- 構成ファイルやデータディレクトリに影響を与えることなく、Debian または Ubuntu ホストから PEM サーバーまたはエージェントをアンインストールするには、次のコマンドを呼び出します。

```
apt-get remove package_name
```

- PEM サーバーまたはエージェントを構成ファイルおよびデータディレクトリとともにアンインストールするには、次のコマンドを呼び出します。

```
apt-get purge package_name
```

package_name は、削除するパッケージの名前です。

#### 6.7.1.3 SLES ホストからの PEM コンポーネントのアンインストール

SLES ホストから PEM サーバーまたはエージェントをアンインストールするには、次のコマンドを呼び出します。

```
zypper remove package_name
```

package_name は、削除するパッケージの名前です。

### 6.7.2 Uninstalling Postgres Enterprise Manager components on Windows

ホストから PEM サーバーをアンインストールすると、同じホストにインストールされている PEM エージェントがアンインストールされます。ただし、PEM エージェントをアンインストールしても、同じホストにインストールされている PEM サーバーはアンインストールされません。

Windows Add/Remove Programs アプリケーションを使用して、Windows ホストから PEM コンポーネントを削除できます。 Windows Control Panel から Add/Remove Programs オプションを選択します。control panel が開いたら、プログラムリストで PEM コンポーネントの名前を見つけます。Remove ボタンをクリックして、コンポーネントを削除します。

次の場所にあるアンインストーラーを呼び出すこともできます。

PEM サーバーの場合、C:\Program Files\edb\pem\server\uninstall-pemserver

PEM エージェントの場合 C:\Program Files\edb\pem\agent\uninstall-pemagent

## 6.8 Troubleshooting

### 6.8.1 既知のインストールと構成の問題

#### 6.8.1.1 既存の mod_wsgi との競合

システムに mod_wsgi の競合するバージョンがある場合、RHEL、Rocky、Alma または Oracle Linux に PEM サーバーのインストール中に次のエラーが表示されます。

```
[root@etpgxlt firstuser]# dnf install edb-pem
Updating Subscription Management repositories.
Last metadata expiration check: 0:01:33 ago on Wed 30 Mar 2022 01:28:16 AM EDT.
Error:
 Problem: problem with installed package python3-mod_wsgi-4.6.4-4.el8.s390x
  - package python39-mod_wsgi-4.7.1-4.module+el8.4.0+9822+20bf1249.s390x conflicts␣
↪with python3-mod_wsgi provided by python3-mod_wsgi-4.6.4-4.el8.s390x
  - package python39-mod_wsgi-4.7.1-4.module+el8.4.0+9822+20bf1249.s390x conflicts␣
↪with python3-mod_wsgi provided by python3-mod_wsgi-4.6.4-3.el8.s390x
  - package edb-pem-server-8.4.0-7.rhel8.s390x requires python39-mod_wsgi >= 4.7, but␣
↪none of the providers can be installed
  - package edb-pem-8.4.0-7.rhel8.s390x requires edb-pem-server = 8.4.0-7.rhel8, but␣
↪none of the providers can be installed
  - cannot install the best candidate for the job
(try to add --allowerasing to command line to replace conflicting packages or --skip-
↪broken to skip uninstallable packages or --nobest to use not only best candidate␣
↪packages)
[root@etpgxlt firstuser]#
```

この問題を解決するには、python3-mod_wsgi パッケージを削除し、PEM のインストールを再試行します。

```
dnf remove python3-mod_wsgi
```

#### 6.8.1.2 JIT からのエラー

PEM バックエンド Postgres インスタンスで JIT が有効になっている場合、RHEL、Rocky、Alma または Oracle Linux で PEM サーバーを構成した後、ワーカー.log にこのエラーが表示される場合があります。

```
Tue Nov 28 03:02:19 2023 WARNING: Error clearing zombies: ERROR:  failed to JIT␣
↪module: Added modules have incompatible data layouts: E-m:e-i1:8:16-i8:8:16-i64:64-
↪f128:64-a:8:16-n32:64 (module) vs E-m:e-i1:8:16-i8:8:16-i64:64-f128:64-v128:64-
↪a:8:16-n32:64 (jit)
CONTEXT:  SQL statement "WITH running_agent_job AS (
        SELECT j.jobid, j.agent_id
        FROM pem.job j LEFT JOIN pem.joblog jl ON (j.jobid = jl.jlgjobid)
        WHERE jl.jlgstatus = r AND agent_id = $1 AND j.jobarid != agent_runtime_id
        ORDER BY j.jobid, jl.jlgjobid
        FOR UPDATE SKIP LOCKED
    ), joblog_status_update AS (
        UPDATE pem.joblog jl SET jlgstatus=d
        FROM running_agent_job r
        WHERE r.jobid = jl.jlgjobid AND jl.jlgstatus=r
        RETURNING r.agent_id, jl.jlgjobid, jl.jlgid
    ), jobsteplog_status_update AS (
        UPDATE pem.jobsteplog js SET jslstatus=d
        FROM joblog_status_update jl
        WHERE js.jsljlgid = jl.jlgid AND js.jslstatus=r
        RETURNING jl.agent_id, jl.jlgjobid AS job_id
    )
    UPDATE pem.job j SET jobprocessid=NULL, jobnextrun=NULL, jobarid=NULL
    FROM (SELECT DISTINCT agent_id, job_id FROM jobsteplog_status_update) js
    WHERE js.job_id = j.jobid"
PL/pgSQL function pem.clear_job_zombies(integer) line 8 at SQL statement
```

エラーを解決するには、バックエンドデータベースサーバーの `postgresql.conf` ファイルで `jit` パラメーターを `off` に設定します。

```
jit=off
```

バックエンドデータベースサーバーを再起動します。

#### 6.8.1.3 RHEL Python 3.9 標準ライブラリのバグによるクラッシュ

RHEL 9、Rocky 9、Alma 9 または Oracle Linux 9 サーバーが影響を受ける Python 3.9 バージョンを実行している場合、PEM 構成中に次のエラーが表示されます。

```
TypeError: realpath() got an unexpected keyword argument strict
```

最新の Python 3.9 パッケージに更新することにより、エラーを解決できます。この問題は、2025 年 10 月の時点で Python 3.9.21 の最新リリースに存在しないことを確認しました。ただし、3.9.21 の一部の以前のリリースおよび以前の Python バージョンには存在していました。

```
dnf upgrade python3
```

PEM 構成を再試行します。

#### 6.8.1.4 Apache HTTPD での TRACK / TRACE メソッドのブロック

!!!注バージョンと構成の互換性この構成は、次の PEM のバージョンでデフォルトで事前構成されています。

- PEM 9 - バージョン 9.8.1 以降
- PEM 10 - バージョン 10.3.0 以降

上記にリストされているバージョンより前のバージョンでは、構成文字列はファイル内に存在しますが、正しく配置されません。これを有効にするには、PEM HTTPS エンドポイントを制御する特定の `VirtualHost` ブロックに手動で移動する必要があります。

```
<VirtualHost _default_:8443>
    # Rewrite rule for disable the request
    <IfModule mod_rewrite.c>
        RewriteEngine on
        RewriteCond %{REQUEST_METHOD} ^(TRACE|TRACK|OPTIONS)
        RewriteRule .* - [F]
    </IfModule>
</VirtualHost>
```

- Ubuntu、Debian、または SLES15 上の Apache の場合、`/etc/apache2/sites-enabled/edb-ssl-pem.conf` ファイルを更新します。
- RHEL での HTTPD の場合、 `/etc/httpd/conf.d/edb-ssl-pem.conf` ファイルを更新します。

### 6.8.2 PEM サーバーの再構成

状況によっては、PEM サーバーのアンインストール、再インストール、およびサーバーの再構成が必要になる場合があります。そのためには

1. PEM サーバー構成を削除し、アンインストールします。

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

2. PEM パッケージを削除します。

```
yum erase edb-pem-server
```

3. pem データベースを削除します。

```
DROP DATABASE pem
```

4. 証明書を`/root/.pem/` から別の場所に移動します。

```
mv /root/.pem/* <new_location>
```

5. agent.cfg ファイルを/usr/edb/pem/agent/etc/agent.cfg から別の場所に移動します。

```
mv /usr/edb/pem/agent/etc/agent.cfg <new_location>
```

6. 構成データディレクトリを別の場所に移動します。

```
mv /var/lib/pemhome/.pem <new_location>
```

7. 次に、PEM サーバーを再度構成します。

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

### 6.8.3 PEM Web アプリケーションがロードされません

PEM Web アプリケーションがロードされていない場合は、 *check the web services are running* 。実行中の場合は、Web サーバーとアプリケーションエラーログを確認します。

Web サーバーとして NGINX を使用している場合、Web サーバーログは/var/log/nginx/error.log にあり、アプリケーションログは/var/log/edb-uwsgi/pem.log にあります。

Apache HTTPD を使用している場合 ログは/var/log/httpd/error_log にあります。このファイルには、Web サーバーとアプリケーションのエラーメッセージの両方が含まれています。

Web サーバーとして Apache HTTPD を使用しており、ログに次のメッセージが表示される場合。

```
Truncated or oversized response headers received from daemon process edbpem: /usr/edb/
↪pem/web/pem.wsgi
```

/etc/httpd/conf フォルダーにある Apache httpd.conf ファイルの下部に次のステートメントを追加します。

```
WSGIApplicationGroup %{GLOBAL}
```

次のステートメントを追加した後、HTTPD サーバーを再起動します。

```
sudo systemctl restart httpd.service
```

### 6.8.4 PostgreSQL サーバーへの接続エラー

PostgreSQL サーバーに接続すると、次のいずれかのエラーメッセージが表示される場合があります。メッセージを注意深く確認します。各エラーメッセージは、問題を解決するために必要な情報を取り入れようとします。

- **サーバーへの接続が失われました**このエラーメッセージは、接続の試行が指定されたしきい値よりも時間がかかったことを示します。[サーバー] ダイアログボックスに表示される接続プロパティに問題があるか、ネットワーク接続の問題、またはサーバーが実行されていない可能性があります。

- **サーバーに接続できませんでした 接続は拒否されました**このエラーには 2 つの理由が考えられます。
- データベースサーバーは実行されていません。サーバーを起動します。
- サーバーは、表示されているアドレスで TCP/IP 要求を受け入れるように構成されていません。

セキュリティ上の理由から、PostgreSQL サーバーは「そのままでは」TCP/IP ポートでリッスンしません。代わりに、TCP/IP 要求をリッスンするように有効にする必要があります。バージョン 7.3.x および 7.4.x の `postgresql.conf` ファイルに `tcpip = true` を追加します。バージョン 8.0.x 以降に `listen_addresses='*'` を追加します。これらの追加により、サーバーは任意の IP インターフェイスでの接続を受け入れます。

詳細については、runtime configuration に関する PostgreSQL ドキュメントを参照してください。

- **FATAL pg_hba.conf エントリー**接続時に PEM がこのメッセージを表示する場合、ネットワーク経由でサーバーに正しく接続できますが、接続を受け入れるように構成されていません。クライアントは、データベースの合法的なユーザーとして検出されませんでした。

サーバーに接続するには、PEM クライアントのホストからの接続を受け入れるようにデータベースサーバー上の `pg_hba.conf` ファイルを構成します。データベースサーバホスト上の `pg_hba.conf` ファイルを変更し、次のフォームでエントリを追加します。

- `host template1 postgres 192.168.0.0/24 md5` IPV4 ネットワークの場合
- `host template1 postgres ::ffff:192.168.0.0/120 md5` IPV6 ネットワークの場合

詳細については、client authentication に関する PostgreSQL ドキュメントを参照してください。

- **FATAL パスワード認証に失敗しました** `password authentication failed for user` エラーメッセージは、入力したパスワードに問題がある可能性を示しています。パスワードを再試行して、正しく入力したことを確認します。エラーメッセージが返された場合は、正しいパスワードを持っていること、サーバーへのアクセスが許可されていること、およびサーバーの `postgresql.conf` 構成ファイルでアクセスが正しく構成されていることを確認します。

### 6.8.5 PEM Web サーバーの接続性チェック

次のコマンドを実行して、クライアントが PEM サーバーに到達し、応答を取得できるかどうかを確認します。

```
curl https://<SERVER_ADDR>:8443/pem/misc/ping -k -i
__OUTPUT__
PING
```

<SERVER_ADDR> は、PEM サーバーの IP アドレスです。出力 `PING` は、PEM Web サーバーが稼働していることを確認します。

### 6.8.6 SQL エラー-リレーション「server」の列「ssl」が存在しません

特定の PEM エージェントと PEM サーバーバージョン間の非互換性により、pemworker を介したサーバー登録が「欠落列」SQL エラーで失敗する場合があります。

**影響を受けるシナリオ**

- PEM エージェント 10.3 より前の PEM サーバーとペアリングした PEM エージェント 10.3。
- PEM サーバー 10.1.1 とペアになった PEM エージェント 10.1.1 注-バージョン 10.1.2 は影響を受けません。

この問題は、中央サーバーの前にエージェントが更新される段階的なアップグレード中に最も頻繁に発生します。

**推奨解像度**

- アップグレードを完了する - 新しいサーバーを登録する前に、PEM サーバーとエージェントの両方が一致するバージョン 10.3 以降に更新されていることを確認します。
- Web UI を使用する - コマンドラインの代わりに PEM Web インターフェイスを介してサーバーを登録します。
- 一時的なダウングレード - PEM エージェントを古いバージョンに戻して登録を実行し、エージェントのアップグレードを続行します。

### 6.8.7 SSL エラー-不完全な sslutils のアップグレード

sslutils システムパッケージがバージョン 1.4 にアップグレードされるが、対応するデータベース拡張機能が以前のバージョンのままである場合、エージェント登録は次の OpenSSL エラー `Error_setting_validity_before_time` で失敗します。

- **解決策** - データベース拡張機能をシステムパッケージと同期するには、PEM バックエンドデータベースで次のコマンドを実行します。

```
ALTER EXTENSION sslutils UPDATE TO 1.4;
```

### 6.8.8 Nginx SSL_read() 失敗エラー

Nginx エラーログに次のエラーが表示された場合、サーバーが応答を送信する前にクライアントが接続を閉じたことを示しています。これは通常、クライアント側の問題であり、PEM サーバー自分自身の問題ではありません。

```
SSL_read() failed (SSL: error:0A000126:SSL routines::unexpected eof while reading)␣
↪while keepalive
```

**原因**

この問題は、クライアントが PEM サーバーへの要求を開始したが、`close_notify` アラートを送信せずに TLS 接続を閉じた場合に発生します。Nginx エラーログは、これを、読み取り中の予期しないファイルの終わり EOF による SSL 読み取り失敗としてキャプチャします。

次の環境に影響します。

- OpenSSL バージョンは 1.1.1e または 3.0.x です
- Nginx バージョンが 1.21.2 より前である

#### 解像度

Nginx をバージョン 1.21.2 以降にアップグレードします。これには、この問題の修正が含まれています。Linux ディストリビューションが提供する Nginx パッケージを使用している場合、このバージョンを含む更新を確認してください。

この例は、EL9 ベースのシステムに互換性のある Nginx バージョン 1.24.0 をインストールし、サービスをリスタートする方法を示しています。

```
#  Install newer Nginx and related packages

dnf install -y --allowerasing \
  https://pkgs.sysadmins.ws/el9/modular/x86_64/nginx/nginx-cp-1.24.0-1.el9.x86_64.rpm␣
↪\
  https://pkgs.sysadmins.ws/el9/modular/x86_64/nginx/nginx-cp-filesystem-1.24.0-1.el9.
↪noarch.rpm \
  https://pkgs.sysadmins.ws/el9/modular/x86_64/nginx/lua-resty-core-0.1.26-1.el9.
↪noarch.rpm  \
  https://pkgs.sysadmins.ws/el9/modular/x86_64/nginx/lua-resty-lrucache-0.13-1.el9.
↪noarch.rpm \
  https://pkgs.sysadmins.ws/el9/base/x86_64/luajit2-2.1.0-20220915.2.el9.0.1.x86_64.
↪rpm

#  Restart Nginx to apply changes

systemctl restart nginx
```

Nginx をアップグレードして再起動した後、クライアントが接続を途中で閉じたときに SSL_read() 障害エラーは発生しません。

# 第7章　ADMINISTERING

## 7.1 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。Web サーバー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により、構成後にポートを変更できます。これらのファイルの名前と場所はプラットフォーム固有です。

### 7.1.1 RHEL の場合

#### 7.1.1.1 SELinux 構成を変更する

デフォルトでは、SELinux は RHEL で有効になっています。設定されたポート番号へのアクセスが許可されていることを確認します。

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
```

2. 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
```

3. 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>
```

#### 7.1.1.2 Web サーバー NGINX の構成

1. ファイル/etc/nginx/conf.d/edb-pem.conf を編集し、次のセクションの 8443 をポート番号に置き換えます。

```
server {
   listen  80;
   listen  [::]:80;
   server_name  HTTPD-EDBPEM-SERVER-v10;

   location /pem/ {
      return 301 https://$host:8443$request_uri;
   }
}
```

(次のページに続く)

(前のページからの続き)

```
server {
   listen       8443 ssl;
   listen       [::]:8443 ssl;
   server_name  HTTPD-EDBPEM-SERVER-v10;

   ssl_certificate /usr/edb/pem/resources/server-pem.crt;
   ssl_certificate_key /usr/edb/pem/resources/server-pem.key;
   ssl_protocols TLSv1.2 TLSv1.3;
   ssl_prefer_server_ciphers on;
   ssl_ciphers EECDH+AESGCM:EDH+AESGCM;
   ssl_ecdh_curve secp384r1;

   location /pem/ {
      include uwsgi_params;
      uwsgi_pass unix:/var/run/edb-uwsgi/pem.socket;
   }
}
```

1. nginx サービスを再起動します。

```
sudo systemctl restart nginx
```

#### 7.1.1.3 Web サーバー Apache HTTPD の構成

1. ファイル/etc/httpd/conf.d/edb-ssl-pem.conf を編集し、次のパラメーターの 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. 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。

### 7.1.2 Debian および Ubuntu の場合

#### 7.1.2.1 Web サーバー NGINX の構成

1. ファイル/etc/nginx/sites-available/edb-pem.conf を編集し、次のセクションの 8443 をポート番号に置き換えます。

```
server {
   listen  80;
   listen  [::]:80;
   server_name  HTTPD-EDBPEM-SERVER-v10;

   location /pem/ {
      return 301 https://$host:8443$request_uri;
   }
}

server {
   listen       8443 ssl;
   listen       [::]:8443 ssl;
   server_name  HTTPD-EDBPEM-SERVER-v10;

   ssl_certificate /usr/edb/pem/resources/server-pem.crt;
   ssl_certificate_key /usr/edb/pem/resources/server-pem.key;
   ssl_protocols TLSv1.2 TLSv1.3;
   ssl_prefer_server_ciphers on;
   ssl_ciphers EECDH+AESGCM:EDH+AESGCM;
   ssl_ecdh_curve secp384r1;

   location /pem/ {
      include uwsgi_params;
      uwsgi_pass unix:/var/run/edb-uwsgi/pem.socket;
   }
}
```

1. nginx サービスを再起動します。

```
sudo systemctl restart apache2
```

#### 7.1.2.2 Web サーバー Apache HTTPD の構成

1. ファイル/etc/apache2/sites-available/edb-ssl-pem.conf を編集し、次のパラメーターの 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。

## 7.2 Registering a PEM agent

PEM エージェントを使用する前に、PEM サーバーに登録する必要があります。**PEM サーバーパッケージによってインストールされた PEM エージェントは、サーバーの構成中に自動的に登録されます。**他のすべてのエージェントの場合、これらの手順に従ってください。

> **注釈**
>
> 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。再度登録する必要はありません。

### 7.2.1 PEM エージェントを登録する方法

Linux および Windows ホストでは、PEM エージェントパッケージには pemworker と呼ばれるコマンドラインユーティリティが含まれています。これを使用して、*pemworker* **ユーティリティを使用した** *PEM* **エージェントの登録** を含む管理タスクを実行できます。

Windows では、PEM エージェントグラフィカルインストーラーを使用して、インストール時にエージェントを登録できます。この便利なオプションは、pemworker ユーティリティが提供するすべての可能性をサポートしているわけではありません。インストーラーにエージェントを登録したくない場合は、**Register now** チェックボックスをオフにします。詳細については、installation instructions を参照してください。

### 7.2.2 pemworker ユーティリティを使用した PEM エージェントの登録

pemworker ユーティリティは、PEM エージェントとともにインストールされます。これは、Linux では `/usr/edb/pem/agent/bin` 、Windows では `C:\Program Files\edb\pem\agent-x64\bin` にあります。

エージェントを登録するには、PEM サーバーのパスワードを設定し、例に示すようにユーティリティを呼び出し、必要に応じてテーブルから関連オプションを追加します。各オプションの後には、対応する値を続けます。

#### 7.2.2.1 Linux

```
export PEM_SERVER_PASSWORD=edb

#  Running as root

pemworker --register-agent
```

#### 7.2.2.2 Windows

```
set PEM_SERVER_PASSWORD=edb

#  Running as admin

./pemworker.exe REGISTER
```

| Option | Description |
|---|---|
| *--pem-server* | PEM バックエンドデータベースサーバーの IP アドレス。`--pem-registration-conn-string`を指定しない限り、このパラメーターは必須です。 |
| *--pem-port* | 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。 |
| *--pem-user* | pem_admin ロールと PEM バックエンドデータベースサーバーに`rolcreaterole`フラグが設定されているデータベースユーザーまたはスーパーユーザーの名前。`--pem-agent-user`またはエージェント接続文字列を介して既存のユーザーを指定する場合、後者は必要ありません。このユーザーは PEM サーバーに接続し、エージェント登録を実行するために使用されます。`--pem-registration-conn-string`を指定しない限り、このパラメーターは必須です。 |
| *--pem-agent-user* | 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。登録後、エージェントはこのユーザーを使用して PEM データベースサーバーへの接続を開き、プローブデータの書き込み、アラートの評価などを行います。このパラメーターはオプショナル。省略すると、エージェントは登録時に作成された`agent<N>`という名前の新しいユーザーを使用して接続します。`<N>`はエージェント ID です。 |
| *--pem-ssl-mode* | 使用する PEM エージェントユーザーの *Deciding on an SSL model* 上記を参照してください。使用可能な値は`prefer`、*require*、*disable*、*verify-CA*、および`verify-full`です。デフォルト値は`require`です。 |
| *--pem-registration-conn-string* | 登録中に PEM サーバーに接続するために使用される接続文字列。 **接続文字列を使用する** を参照してください。 |
| *--cert-path* | 証明書を保存するディレクトリへの完全なパス。パスを提供しない場合、証明書は Linux では`~/.pem`、Windows では`%APPDATA%/pem`で作成されます。 |
| *--config-dir* | 構成ファイルのディレクトリパス。デフォルトは`<pemworker path>/../etc`です。 |
| *--display-name* | PEM ブラウザツリーに表示されるエージェントのわかりやすい名前。デフォルトはホストの完全修飾ドメイン名 FQDN であり、このオプションが設定されていない場合はホスト名にフォールバックします。 |
| *--force-registration* | `force-registration`句を含めて、指定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`Yes`です。 |
| *--group* | エージェントを配置するグループの名前。このパラメーターはオプショナル。省略すると、エージェントはグループに配置されません。 |
| *--team* | 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。このエージェントへのアクセスは、名前付けたロール、所有者、および pem_super_admin ロールのみに制限されています。このパラメーターはオプショナル。省略すると、チームは割り当てられず、すべてのユーザーがこのエージェントにアクセスできます。 |
| *--owner* | 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。このユーザーは、エージェントの所有者として割り当てられます。省略すると、指定された`pem-user`が所有者として割り当てられます。 |

次のページに続く

表 1 – 前のページからの続き

| Option | Description |
|---|---|
| *--cluster-name* | エージェントオブジェクトを追加するオブジェクトエクスプローラーでクラスター名前を指定します。クラスターが存在しない場合、自動的に作成されます。このパラメーターはオプショナル。 |
| *--profile-id* | このエージェントに割り当てる既存のエージェントプロファイルの ID を指定します。オプショナル。 |
| *--profile-name* | このエージェントに割り当てる既存のエージェントプロファイルの名前を指定します。このパラメーターはオプショナルであり、設定されている場合 `--profile-id` によってオーバーライドされます。 |
| *--allow_server_restart* | PEM が監視対象サーバーを再起動できます。デフォルト値は `True` です。 |
| *--allow-batch-probes* | PEM がこのエージェントでバッチプローブを実行できるようにします。デフォルト値は `False` です。 |
| *--batch-script-user* | バッチ/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 none です。このパラメーターを空白のままにするか、指定され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。 |
| *--enable-heartbeat-connection* | PEM エージェントとサーバーの間に専用のハートビート接続を作成して、アクティブステータスを更新します。デフォルト値は `False` です。 |
| *--enable-smtp* | 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 `False` です。 |
| *--enable-snmp* | 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 `False` です。 |
| *--enable-webhook* | PEM エージェントが PEM サーバーに代わって Webhook エンドポイントを呼び出すことができます。デフォルト値は `False` です。 |
| *-o* | 構成ファイルオプションをオーバーライドします。使用方法については、次の デフォルト構成のオーバーライド-例 を参照してください。 |

**注釈**

*agent.cfg* ファイルから `pem-agent-user` パラメーターを手動で削除すると、エージェントはデフォルトユーザーを使用して接続しません。

#### 7.2.2.3 pemworker ユーティリティの認証

PEM データベースに変更を加える前に、接続は 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に、PEM サーバーの管理ユーザーロール postgres に関連付けられたパスワードを提供する必要があります。管理パスワードは 3 つの方法で指定できます。

- `PEM_SERVER_PASSWORD` 環境変数を設定します。
- `PGPASSWORD` キーワードを使用して、コマンドラインにパスワードを入力します。
- `.pgpass` ファイルにエントリーを作成します。
- パスワードまたは暗号化されたパスワードを`--pem-registration-conn-string` に含めます

パスワードを入力しない場合、パスワード認証エラーが発生します。認証が成功すると、その他の不足している必須情報の入力を求められます。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。

#### 7.2.2.4 PEM エージェントサービスを有効にする

エージェントは登録されましたが、まだ実行されていません。エージェントの実行を開始し、システムのリブート後も実行されたままであることを確認するには、PEM エージェントサービスを有効にする必要があります。

#### 7.2.2.5 Linux

```
systemctl enable --now pemagent
```

#### 7.2.2.6 Windows

Windows サービスマネージャー services.msc で pemAgent サービスを見つけ、右クリックして **Restart** を選択します。 **Startup Type** が **Automatic** に設定されていることを確認します。

### 7.2.3 PEM エージェント登録の検証

`verify-registration` サブコマンドは、PEM 10.4 で導入された診断ユーティリティです。これにより、管理者は、PEM エージェントが適切に登録され、PEM データベースでアクティブで、PEM サーバーに接続できることを確認できます。

このコマンドは、次のチェックを実行して、エージェントの動作ステータスを検証します。

1. **構成の完全性**ローカル構成に適切な認識可能なエージェント ID が含まれていることを検証します。
2. **安全な接続** `agent.cfg` ファイルで構成された証明書ベースの認証を使用して、PEM サーバーへの接続を確立できる機能を検証します。
3. **データベースステータス** エージェントが PEM データベースに**アクティブ**として登録され、明示的にマークされていることを確認します。

このサブコマンドは、次の場合に特に役立ちます。

- **登録後の検証**初期登録の直後にエージェントのセットアップを検証します。
- **接続のトラブルシューティング**エージェントと PEM サーバー間の接続の問題の診断。
- **SSL/証明書監査** SSL 証明書が適切に構成され有効であることを確認します。
- **ヘルスモニタリング**自動ヘルスチェックとモニタリングスクリプトでのエージェントステータスの確認。

#### 7.2.3.1 コマンド構文

verify-registration サブコマンドの構文は、各オペレーティングシステムに固有です。

**Linux**

```
pemworker --verify-registration [verify-registration-options]
```

**Windows**

```
pemworker VERIFY-REGISTRATION [verify-registration-options]
```

| Option | Description |
|---|---|
| *--config-dir <path>* | エージェント構成ファイル *agent.cfg* が存在するディレクトリパス。指定しない場合、デフォルトの構成ディレクトリが使用されます Linux の `/usr/edb/pem/agent/etc/` など。 |

コマンドが実行されると、 pemworker ユーティリティは 4 段階の検証プロセスを実行します。

1. **ロード構成** agent.cfg デフォルトまたは指定されたパスからを読み取り、エージェント ID と接続文字列を取得します。
2. **接続の確立**登録時に定義された証明書ベースの SSL 認証を使用して、PEM データベースサーバーへの接続を開こうとします。
3. **クエリステータス** pem.agent システムテーブルに対してチェックを実行して、ID が存在し、ステータスがアクティブに設定されていることを確認します。
4. **レポート結果**成功または失敗メッセージをコンソールに出力し、特定の終了コードを返します。

#### 7.2.3.2 終了コードの理解

コマンドは次の終了コードを返します。

| Code | Status | Description |
|---|---|---|
| 0 | Success | エージェントは登録され、アクティブで、接続されています。 |
| 1 | Config Error | ローカル構成からの有効なエージェント ID の取得に失敗しました。 |
| 2 | Connection Error | PEM サーバーへの接続を確立できません。 |

次のページに続く

表 3 – 前のページからの続き

| Code | Status | Description |
|---|---|---|
| 3 | Database Error | エージェント ID が見つかりましたが、PEM データベースで非アクティブとしてマークされています。 |

#### 7.2.3.3 使用例

**例 1 基本的な検証**

デフォルト設定を使用して登録を確認するには

**Linux**

```
pemworker --verify-registration
```

**Windows**

```
pemworker VERIFY-REGISTRATION
```

**成功した場合の予想出力**

```
Verify registration done successfully.
```

**例 2 カスタム構成ディレクトリを使用する**

デフォルト以外の構成ディレクトリを使用するエージェントを確認するには

**Linux**

```
pemworker --verify-registration --config-dir /opt/custom/pem/agent/etc
```

**Windows**

```
pemworker VERIFY-REGISTRATION --config-dir "C:\custom\pem\agent\etc"
```

### 7.2.4 PEM エージェントの登録解除

pemworker ユーティリティを使用して、PEM エージェントの登録を解除できます。エージェントの登録を解除するには、以下の例に示すように pemworker ユーティリティを呼び出します。

**Linux**

```
#  Running as root

pemworker --unregister-agent
```

**Windows**

```
./pemworker.exe UNREGISTER-AGENT
```

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

| Option | Description |
|---|---|
| *--pem-user <username>* | 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。このパラメーターは必須です。 |
| *--config-dir* | 構成ファイルのディレクトリパス。デフォルトは`"<pemworker path>/../etc"`です。 |

### 7.2.5 高度な使用法

以下は、PEM エージェント登録の詳細なオプションの一部です。

#### 7.2.5.1 接続文字列を使用する

PEM エージェントが PEM バックエンドデータベースに接続する方法をより詳細に制御するために、接続文字列を指定できます。指定できる接続文字列は、*registration* **接続文字列と** agent*接続文字列です。これらの接続文字列は、キー値または URI 形式であり、libpq でサポートされているオプションを含みます。ただし、`pem` にハードコードされた `dbname` と、`read-write` にハードコードされた `target_session_attrs` は除きます。

#### 7.2.5.2 登録接続文字列

`--pem-registration-conn-string` を使用して指定される登録接続文字列は、PEM バックエンドへの「ワンタイム」特権接続に使用され、次のタスクを実行します。

- エージェント登録 –register-agent
- エージェント登録解除 –unregister-agent
- サーバー登録 –register-server
- サーバーの登録解除 –unregister-server

登録文字列は、これらのアクションを実行するために必要な権限を持つユーザーを指定する必要があります、特に

- `pem_admin` ロール
- `rolcreaterole` `--pem-agent-user` またはエージェント接続文字列を介して既存のユーザーも指定しない限り下記を参照してください。

登録接続文字列自分自身は保存されないため、これらのタスクのいずれかを実行するために使用する場合は常に入力する必要があります。ホストとポートは、エージェントの登録に使用される接続文字列から抽出され、agent.cfg ファイルに保存されるため、エージェントは PEM バックエンドの場所を知ることができます。

環境変数 PEM_REGISTRATION_CONN_STRING を使用して登録接続文字列を指定することもできます。優先順位は次のとおりです。

1. コマンドライン引数 –pem-registration-conn-string - 最も高い優先度
2. 環境変数 PEM_REGISTRATION_CONN_STRING
3. 個々のパラメーター –pem-server、–pem-port、–pem-user - 最低優先度

#### 7.2.5.3 エージェント接続文字列

エージェント接続文字列は、エージェントが進行中のデータ収集などのために PEM バックエンドに接続するために使用される永続的な接続に使用されます。これはオーバーライドオプションであり、agent.cfg に直接書き込まれるため、`-o agent_connection_string` を使用して指定する必要があります。

エージェント接続文字列が設定されていない場合、エージェントは `agent.cfg` `pem_host` 、`pem_port` などの個々のパラメーターまたは Windows のレジストリを使用して、接続する方法と場所を決定します。

エージェント接続文字列が設定されている場合、これらのパラメーターは無視されます。エージェント登録時にエージェント接続文字列が指定された場合、PEM はエージェント接続のユーザーまたはクライアント証明書を作成しようとしません。接続文字列を指定する場合、指定されたユーザーが存在し、認証可能で、`pem_agent` ロールがあることを確認する必要があります。

エージェント接続文字列は、`agent.cfg` または Windows のレジストリに保存されます。文字列に `password` または `sslpassword` 値が含まれる場合、暗号化され、それぞれカスタムの `pemenc_password` または `pemenc_sslpassword` 値に置き換えられます。特殊文字を含むパスワードは、一重引用符で囲まれるキー-値スタイルまたは URL エンコード URI スタイルする必要があります。

環境変数 PEM_AGENT_CONN_STRING を使用してエージェント接続文字列を指定することもできます。優先順位は次のとおりです。

1. コマンドライン引数 `-oagent_conn_string=...` - 最も高い優先度
2. 環境変数 PEM_AGENT_CONN_STRING
3. 構成ファイル/レジストリ `agent_conn_string` / AGENT_CONN_STRING
4. 個々のパラメーター `pem_host` 、`pem_port` 、`pem_ssl_mode` など - 最低優先度

#### 7.2.5.4 エージェント ID の設定

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。**デフォルト構成のオーバーライド-例** に示すように、 `-o` オプションを使用して値が提供されない限り、値 `max(id)+1` は各エージェント ID に割り当てられます。

#### 7.2.5.5 デフォルト構成のオーバーライド-例

`-o` オプションを使用すると、登録中に設定できないエージェント構成パラメーター Linux では `agent.cfg` および Windows では `HKEY_LOCAL_MACHINE\Software\EnterpriseDB\PEM\agent` に保存されますをオーバーライドできます。これらの入力は通常の CLI オプションと同じ方法では検証されないため、このオプションは注意して使用する必要があります。パラメーターの完全なリストは *Modifying agent configuration* にあります

以下に、-o オプションの使用例をいくつか示します。

#### 7.2.5.6 エージェント ID の設定

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。

```
#  Running as root

/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 が利用可能なため、PEM エージェントは正常に登録されました。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、エラーが発生します。

#### 7.2.5.7 既存の SSL 証明書とキーの提供

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。既存の SSL 証明書とキーファイルを割り当てて、特定のエージェント ID に対して新しい証明書が生成されないようにします。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。<ID> は、コマンドラインを使用して指定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。

```
#  Running as root

#  List the location of valid SSL certificates 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


#  Register the PEM agent using command line. Assign the

#  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 \
```

(次のページに続く)

(前のページからの続き)

```
- -display-name agent5 \
- o agent_id=5 \
- o agent_ssl_crt=/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は、適切な権限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、または適切な権限がない場合、エラーが発生します。

#### 7.2.5.8 非 root ユーザーアカウントを使用して Linux で PEM エージェントを登録する

非 root 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 root ユーザーとして PEM エージェントを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 root ユーザーの ID を引き継ぎます。次に

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。

```
# Running as nonroot user edb
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_crt=/home/edb/pem/agent<id>.crt
```

(次のページに続く)

(前のページからの続き)

```
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、エージェントを起動します。

```
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 ユーザーで起動されました。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。前述のコマンドを使用して手動で起動する必要があります。

5. オプションで、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して、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起動できます。

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

```
# Running as superuser
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-snmp++/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
sudo systemctl enable pemagent
sudo systemctl start pemagent
```

(次のページに続く)

(前のページからの続き)

```
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たとえば、enterprisedb が所有するファイルへのアクセスを必要とするプローブとジョブは失敗します。 - *agent.cfg* ファイルをデフォルトの場所から別の場所に移動すると、PEM ダッシュボードにエージェントステータスが`unknown` として表示される場合があります。詳細については、 *Troubleshooting agent issues* を参照してください。

## 7.3 Registering a Postgres server

PEM でデータベースサーバーを管理またはモニタする前に、データベースサーバーを PEM に登録し、以前に登録した PEM エージェントにデータベースサーバーをバインドする必要があります。データベースサーバーをリモートエージェント別のホストにあるエージェントにバインドできます。ただし、エージェントが別のホストにある場合、インスタンスに関するすべての統計情報にアクセスできません。

### 7.3.1 Postgres サーバーを登録するとどうなりますか?

サーバーを登録すると、次の 2 つのことが行われます。

- PEM Web アプリケーションのサーバーツリーにサーバーを追加し、ユーザーが Web アプリケーションからサーバーへの接続を開くことができます。- サーバーを特定のエージェントにバインドします。つまり、エージェントがサーバーへの接続を開き、プローブとスケジュールされたジョブを実行します。

登録後、エージェントは指定されたデータベースへの接続を開きます。この接続を使用して、同じユーザーで接続できる他のデータベースを判断し、それぞれに 1 つの接続を生成します。

サーバーを登録するには 3 つの方法があります。

- バインドするエージェントが存在するホストのコマンドラインから *pemworker* **ユーティリティを使用してサーバーを登録する**- PEM Web アプリケーションの **自動サーバー検出へのサーバーの登録** を使用して、エージェントと同じホストにある Postgres サーバーを自動的に登録します。- **データベースサーバーの手動登録** PEM Web アプリケーションのすべての詳細を追加します。

### 7.3.2 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、Postgres サーバーを登録できます。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あるエージェントに新しいサーバーをバインドします。

Linux ホストでサーバーを登録するには、次のコマンドを使用します。

```
export PEM_SERVER_PASSWORD=example
export PEM_MONITORED_SERVER_PASSWORD=example
pemworker --register-server
```

Windows ホストにサーバーを登録するには、次のコマンドを使用します。

```
set PEM_SERVER_PASSWORD=example
set PEM_MONITORED_SERVER_PASSWORD=example
pemworker.exe REGISTER-SERVER
```

環境変数 PEM_SERVER_PASSWORD は、指定された--pem-user のパスワードに設定する必要があります。必要に応じて、このユーザーに *pemworker* **接続の追加パラメーターの指定** を実行できます。環境変数 PEM_MONITORED_SERVER_PASSWORD を、エージェントサーバーバインドユーザーのパスワードに設定します前者が指定されていない場合は--asb-host-user または--server-user 。

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

| Option | Description |
|---|---|
| *--pem-user* | サーバーの詳細を PEM データベースに書き込むために使用する PEM サーバー上の PEM 管理ユーザー `pem_admin`ロールが必要です。必須。 |
| **Server parameters** | これらのパラメーターは、PEM Web アプリケーションに接続プロパティを設定するために使用されます。これらは、ユーザーが PEM Web アプリケーションから監視対象サーバーに接続するときに使用されます。これらは、エージェントサーバーバインディングパラメーターでオーバーライドされない限り、エージェントから監視対象サーバーへの接続にも使用されますこの表の **エージェントサーバーバインディングパラメーター** を参照してください。 |
| *--server-addr* | 監視対象サーバーの IP アドレスまたは完全修飾ドメイン名を指定します。Linux システムでは、アドレスフィールドを空白のままにして、ローカルマシン上のデフォルトの PostgreSQL Unix ドメインソケットを使用できます。または、PostgreSQL ソケットへの代替パスに設定できます。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。この場合、使用されませんが、`PEM_MONITORED_SERVER_PASSWORD`を空以外の値に設定する必要があります。ソケットはローカルでのみアクセスできるため、ソケット接続を使用する場合は、代わりに`asb-host-name`を介してこれを指定することをお勧めします。必須。 |
| *--server-port* | 監視対象サーバーのポート番号を指定します。必須。 |
| *--server-database* | 接続する監視対象サーバー上のデータベースの名前を指定します。必須。 |
| *--server-user* | 監視対象サーバーへの接続に使用するユーザーの名前を指定します。必須。 |
| *--server-service-name* | 監視対象の Postgres サーバー、たとえば、Linux の systemd サービスユニットを管理するオペレーティングシステムサービスの名前を指定します。オプショナル。 |
| **Agent-server binding parameters** | これらのパラメーターを使用して、サーバーパラメーターをオーバーライドします。 PEM Web アプリケーションで使用される資格情報とは異なる資格情報を使用してエージェントを監視対象サーバーに接続する場合、これらを使用します。 |

次のページに続く

表 5 – 前のページからの続き

| Option | Description |
|---|---|
| *--asb-host-name* | 監視対象サーバーの IP アドレスまたは完全修飾ドメイン名を指定します。また、Linux システムでは、PostgreSQL ソケットへのパスを提供できます。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。この場合、使用されませんが、 *PEM_MONITORED_SERVER_PASSWORD* `を空以外の値に設定する必要があります。オプショナル。指定しない場合、デフォルトの`--server-addr`。 |
| *--asb-host-port* | 監視対象サーバーのポート番号を指定します。オプショナル。指定しない場合、デフォルトの`--server-port`。 |
| *--asb-host-db* | エージェントが接続する監視対象サーバー上のデータベースの名前を指定します。オプショナル。指定しない場合、デフォルトの`--server-database`。 |
| *--asb-host-user* | エージェントが監視対象サーバーに接続するために使用するユーザーの名前を指定します。オプショナル。指定しない場合、デフォルトの`--server-user`。 |
| *--asb-ssl-mode* | 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*、および`verify-full`が含まれます。オプション、デフォルトは`prefer`です。 |
| **Server metadata** | これらのパラメーターは、PEM Web アプリケーションでサーバーがどのように表示されるかを決定します。 |
| *--group* | サーバーが表示されるグループの名前を指定します。オプショナル。 |
| *--team* | 監視対象サーバーが属するチームとして割り当てられる PEM サーバー上の Postgres ロールを指定します。このロールを持つユーザーのみがサーバーにアクセスできます。オプショナル。デフォルトは none で、すべてのユーザーがアクセスできます。 |
| *--owner* | 監視対象サーバーの所有者として割り当てる PEM サーバー上の Postgres ユーザーを指定します。オプション、デフォルトは`--pem-user`です。 |
| *--display-name <name>* | 監視対象データベースサーバーの表示名を指定します。オプショナル、デフォルトはシステムホスト名です。 |
| *--cluster-name* | サーバーオブジェクトを追加するオブジェクトエクスプローラーでクラスター名前を指定します。クラスターが存在しない場合、自動的に作成されます。オプショナル。 |
| *--profile-id* | このサーバーに割り当てる既存のサーバープロファイルの ID を指定します。オプショナル。 |
| *--profile-name* | このサーバーに割り当てる既存のサーバープロファイルの名前を指定します。このパラメーターはオプショナルであり、設定されている場合`--profile-id`によってオーバーライドされます。 |
| **Cluster management parameters** | これらのパラメーターを使用すると、クラスターマネージャー EDB フェールオーバーマネージャー EFM または Patroni に関する情報を使用している場合、それに関する情報を渡すことができます。 |
| *--replication-solution* | このクラスターで使用されるクラスターマネージャーを指定します。使用可能な値は`efm`および`patroni`です。オプショナル。このオプションを設定する場合、選択したソリューションの次のオプションも設定する必要があります。 |
| *--efm-cluster-name* | サーバーを監視する EDB Failover Manager クラスターの名前を指定します該当する場合。 |

次のページに続く

表 5 – 前のページからの続き

| Option | Description |
|---|---|
| *--efm-install-path* | EDB 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該当する場合。 |
| *--patroni-cluster-name* | サーバーがメンバーである Patroni クラスターの名前を指定します該当する場合。 |
| *--patroni-install-path* | Patroni 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該当する場合。 |
| *--patroni-config-path* | Patroni クラスターの構成ファイルへの完全なパスを指定します該当する場合。 |
| **Other parameters** | NA |
| *--remote-monitoring* | サーバーがエージェントと同じホストにない場合、*yes* `に設定します。リモート監視が有効になっている `*yes*、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利用できず、リモートサーバーは一部の PEM ユーティリティ Audit Manager、Capacity Manager、Log Manager などからアクセスできません。 )。オプション、デフォルトは`no`です。 |
| *--config-dir* | 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。オプション、デフォルトは`<pemworker path>/../etc`です。 |
| *--force-registration* | 指定すると、 `agent.cfg`ファイルが既に存在する場合でも登録は続行されます。このオプションは、不適切に使用するとデータの整合性の問題を発生する可能性があるため、EDB サポートから指示された場合にのみ使用してください。 |

#### 7.3.2.1 pemworker 接続の追加パラメーターの指定

データベースサーバーを登録または登録解除するときは、登録を実行する管理 PEM ユーザー`--pem-user`を指定する必要があります。このユーザーのパスワードを `PEM_SERVER_PASSWORD` として指定する必要もあります。

追加の接続パラメーターを指定する必要がある場合、またはパスワード以外の認証方法を使用する必要がある場合は、以下の手順に従ってください。

pemworker ユーティリティは、libpq を使用して Postgres に接続します。 そのため、libpq environment variables を使用して多くの接続プロパティを設定できます。

例、エージェントの登録に使用される接続の SSL モードを指定する場合

```
export PEM_SERVER_PASSWORD=edb
export PGSSLMODE=require
pemworker --register-agent ...
```

同様に、証明書認証されたアカウントを使用して接続する場合、証明書とキーの詳細を提供できます。`PEM_SERVER_PASSWORD` を提供する必要があることに注意してくださいが、使用されないため、任意の値に設定できます。

```
export PGSSLCERT=pem-server.crt
export PGSSLKEY=pem-server.key
```

(次のページに続く)

(前のページからの続き)

```
export PEM_SERVER_PASSWORD=none
pemworker --register-agent ...
```

#### 7.3.2.2 例

この例では、必要なパラメーターのみを使用してサーバーを登録します。ユーザー admin01 は、PEM サーバーに接続するために使用されます。--server-* パラメーターに指定された資格情報は、エージェントから監視対象サーバーへの接続を作成するために使用されます。同じ詳細は、PEM Web アプリケーションにサーバー接続の詳細を入力するために使用されます。

```
pemworker --register-server \
          --pem-user admin01 \
          --server-addr pg123.prod.infra.business \
          --server-port 5432 \
          --server-database postgres \
          --server-user prod_admin
```

次の例では、エージェントから監視対象サーバーへの接続に使用するさまざまなパラメーターを指定します。これらは、--server-addr で提供された完全修飾ドメイン名を代わりに localhost でオーバーライドします。また、ユーザーをオーバーライドして local_monitor ユーザーを使用します。これは、たとえば、同じホストからのみ接続を許可されたユーザーである場合があります。

この例では、サービス名 postgresql も指定します。これは、PEM エージェントがこのサーバーを再起動して構成変更を適用できることを意味します。

```
pemworker --register-server \
          --pem-user admin01 \
          --server-addr pg123.prod.infra.business \
          --server-port 5432 \
          --server-database postgres \
          --server-user prod_admin \
          --server-service-name postgresql \
          --asb-host-name localhost \
          --asb-host-user local_monitor
```

#### 7.3.2.3 pemworker ユーティリティを使用してサーバーの登録を解除する

pemworker ユーティリティを使用して、データベースサーバーの登録を解除できます。サーバーの登録を解除するには、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-SERVER
```

pemworker ユーティリティを呼び出すときに、コマンド文字列にコマンドラインオプションを追加します。各オプションの後には、対応する値を入力します。

| Option | Description |
|---|---|
| *--pem-user* | PEM 管理ユーザーの名前を指定します。必須。 |
| *--asb-host-name* | エージェントがデータベースサーバーへの接続に使用するホスト名を指定します。 |
| *--server-port* | ホストのポート番号を指定します。必須。 |
| *--config-dir* | 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。オプション、デフォルトは `<pemworker path>/../etc` 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提供し、pemworker ユーティリティが`--pem-user` として PEM サーバーに接続できるようにします。

### 7.3.3 自動サーバー検出へのサーバーの登録

モニタするサーバーがモニタリングエージェントと同じホストにある場合は、[ Auto Discovery ] 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素化できます。

特定のエージェントの自動検出を有効にするには、Server Auto Discovery 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 > Manage Probes** を選択します。 [ **Manage Probes** ] 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。

Auto Discovery 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、**Management > Auto Discovery** を選択します。

[ Auto Discovery ] ダイアログボックスが開くと、[ **Discovered Database Servers** ] 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、**Server Connection Details** ボックスにサーバーに関する情報を表示し、**Agent Connection Details** ボックスにエージェントの接続プロパティを表示します。

**Check All** 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、**Uncheck All** 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クリアします。

**Server Connection Details** ボックスのフィールドは、PEM がモニタするサーバーに関する情報を提供します。

- **Name** フィールドの監視対象サーバーの名前を受け入れるか、変更します。指定した名前は PEM クライアントのツリーに表示されます。
- **Server group** リストを使用して、PEM クライアントツリーでサーバーが表示されるサーバーグループを選択します。
- **Host name/address** 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。

- **Service 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、サービス名を提供する必要があります。
- **Maintenance Database** フィールドは、初期接続とグローバルメンテナンス操作に使用するデータベースを指定します。インストールの**メンテナンスデータベース** フィールドの内容をカスタマイズします。

**Agent Connection Details** 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ディングに使用される IP アドレスが表示されます。
- **User name** 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。
- **Password** フィールドには、指定され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。
- **SSLmode** フィールドを使用して、SSL 接続設定を指定します。

モニタリングにバインドしているサーバーの接続プロパティの指定が終了したら、**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。

### 7.3.4 データベースサーバーの手動登録

PEM でデータベースサーバーを管理またはモニタするには、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。

Register Server 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、PEM クライアントツリーにサーバーを表示できます。[ Create Server ] ダイアログボックスを開くには、 **Object > Register > Server** を選択します。

> **注釈**
>
> - 接続する前に、登録している Postgres サーバーの `pg_hba.conf` ファイルで PEM クライアントのホストからの接続が許可されていることを確認する必要があります。 - データベーススーパーユーザーまたは `pem_admin` ロールを持つユーザーのみがデータベースサーバーを PEM エージェントにバインドできます。

**General** タブを使用して、サーバーの一般的なプロパティを説明します。

- **Name** フィールドを使用して、サーバーの名前を指定します。名前は、PEM ブラウザツリー内のサーバーを識別します。
- **サーバーグループ/クラスター**を使用して、ツリー内のサーバー、エージェント、クラスターを編成できます。サーバーグループ/クラスターを使用すると、多数のサーバーをより簡単に管理できます。たとえば、実稼働グループ、テストグループ、または LAN 固有のグループを持つことができます。**Server group/Cluster** 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/クラスターを選択します。

- **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人、または PEM サーバーのスーパーユーザー権限を持つ PEM ユーザーのみが、PEM にログインする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した場合、デフォルトでは、すべての PEM ユーザーにサーバーが表示されます。[ Server Configuration ] ダイアログボックスの `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。
- **Background** カラーセレクターを使用して、サーバーに保存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、サーバーに保存されているオブジェクトの PEM ツリーのラベルのフォント色を選択します。
- **Save** を選択するときに **Connect now?** を選択してサーバー接続を試行します。PEM クライアントが、後の接続試行まで指定された接続パラメーターを検証したくない場合は、**Connect now?** をクリアします。
- **Comments** フィールドにサーバーに関するメモを入力します。

> **注釈**
>
> Server Properties ダイアログの**Team**フィールドは、次のいずれかの条件が満たされる場合にのみ編集可能です。

- ユーザーは `pem_super_admin` ロールを持っています。
- ユーザーはサーバーの所有者です。
- ユーザーは新しいサーバーを作成していますこれには `pem_database_server_registration` ロールが必要です。

他のすべてのシナリオでは、フィールドは読み取り専用です。

- **Host name/address** フィールドに、サーバホストの 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、ローカルマシンでデフォルトの 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。
- **Maintenance Database** フィールドを使用して、インストールされた pgAgent スキーマと adminpack オブジェクトを含むことが予想される接続する PEM の初期データベースの名前を指定します両方オプショナル。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の混乱を避けるために `postgres` データベースを作成することをお勧めします。
- 監視対象サーバーで Kerberos 認証を使用するには、 **Kerberos Authentication** を **Yes** に設定します。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` から `TRUE` を設定します。

- **Username** フィールドにサーバーで認証するときに使用する名前を指定します。
- 指定され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。
- 後で再利用できるように PEM バックエンドデータベースにパスワードを暗号化された形式で保存するには、**Savepassword?** を選択します。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。保存されたパスワードを削除するには、最初にデータベースサーバーを切断し、次に [ **Object > Clear Saved Password ] を選択します。
- **Role** フィールドを使用して、サーバーに接続した後にクライアントが使用する特権を割り当てられたロールの名前を指定します。この値により、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継ぐことができます。接続するロールは、指定されたロールのメンバーである必要があります。

[ **Parameters** ] タブのフィールドを使用して、接続設定を構成します。

+ ボタンをクリックして、新しいパラメーターを追加します。一般的なパラメーターには次のものが含まれます。

- **ホストアドレス**サーバーの IP アドレスを指定して、DNS ルックアップを回避し、接続速度を向上させます。Kerberos、GSSAPI、SSPI、または verify-full SSL モードを使用する場合、ホスト名と IP アドレスの両方を提供することをお勧めします。
- **パスワードファイル**.pgpass ファイルへのパスを指定して、パスワードなしの認証を有効にします。詳細については、PostgreSQL ドキュメント、セクション 33.15 を参照してください。
- **接続タイムアウト**接続を待機する最大時間秒単位を設定します。値 0 または空は、無期限に待機することを意味します。デフォルトは 10 秒です。2 秒未満の値を使用することはお勧めできません。
- **SSL モード** 使用する SSL 接続の種類を選択します。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。

- **Clientcertificate** 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、デフォルトの<STORAGE_DIR>/<USERNAME>/.postgresql/postgresql.crt を置き換えます。SSL 接続が作成されない場合、このパラメーターは無視されます。
- **Clientcertificate key** 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キー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<STORAGE_DIR>/<USERNAME>/.postgresql/postgresql.key を置き換えます。SSL 接続が作成されない場合、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 接続が作成されない場合、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 にあるデフォルトリストを置き換えます。SSL 接続が作成されない場合、このパラメーターは無視されます。

- **SSLcompression?** 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 接続が作成されない場合、このパラメーターは無視されます。

> **警告**
>
> 証明書、プライベート鍵、および失効リストは、PEM サーバープロセスが実行されるユーザアカウントが所有するサーバー上のユーザーごとのファイル格納領域に格納されます。これは、サーバーの管理者がこれらのファイルにアクセスでき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 Tunnel** 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、中間プロキシホストを介してデータベースサーバーを、クライアントが直接接続できないネットワーク上にあるサーバーに接続できます。

- 指定されたサーバーへの接続時に SSH トンネルを使用するには、 **Use SSH 確かにトンネリング** を **Yes** に設定します。
- **Tunnel host** フィールドに、SSH ホストクライアント接続が転送される名前または IP アドレスを指定します。
- 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 ホストのログイン特権を持つユーザーの名前を指定します。
- **Authentication** フィールドに、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。
  - SSH ホストへの認証にパスワードを使用するには、 **Password** を選択します。これはデフォルトです。
  - 接続時にプライベートキーファイルを使用するには、 **Identity file** を選択します。
- SSH ホストが認証用にプライベートキーファイルを想定している場合、**Identity file** フィールドを使用して、キーファイルの場所を指定します。
- SSH ホストがパスワードを想定している場合は、 **Password** フィールドを使用してパスワードを指定します。ID ファイルを使用している場合、パスフレーズを指定します。
- **Savepassword?** を有効にして、今後使用するためにパスワードを保存するように PEM に指示します。Clear SSH Tunnel Password を使用して、保存したパスワードを削除します。
- **キープアライブ秒**フィールドで秒数を指定して、接続を介してデータが送信されなかった場合、`keepalive` パケットが送信される期間を定義しますリモートホストによって無視されます。これは、NAT 経由で接続を生きたままにするのに役立ちます。`Keep alive` を無効にするには 0 を指定します。

**Advanced** タブを使用して、サーバーの管理に使用される詳細を指定します。

- **DB rotation** 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 制限を指定して、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` および `test_db` データベースのみを表示するには、`'live_db'`,`'test_db'` と入力できます。データベースのプロパティダイアログボックスから pg_namespace に対する制限を入力して、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。

- **Password execcommand** フィールドを使用して、実行時に SQL パスワードを取得するシェルコマンドを指定します。コマンドの stdout はパスワードとして使用されます。

これは、PAM authentication シナリオなど、パスワードが短期間のトークンである場合に役立ちます。

コマンドにプレースホルダーを含めて、接続の詳細を動的に渡すことができます。

- **%HOSTNAME%** – サーバーホスト名
- **%PORT%** –サーバーポート
- **%USERNAME%** – データベースのユーザー名
- **Password exec 有効期限** フィールドを使用して、 **Password exec コマンド** で取得したパスワードの最大有効期間秒単位を設定します。
  - 指定しない場合、パスワードはセッションの間有効なままです。
  - 0 に設定すると、コマンドは新しい接続または再接続のたびに実行されます。
  - パスワードが時間制限されている場合一時トークンなど、認証の失敗を回避するために有効期限の少し前にこの値を設定します。
- **準備しきい値**フィールドを使用して、クエリーを準備する時期を制御します。
  - **0** 最初の実行時にクエリーを準備します。
  - **正の整数**クエリーを準備します。その回数実行された後、クエリーを準備します。days.
  - **空白** この接続では準備されたステートメントを完全に無効にします。

この設定は、準備されたステートメントと互換性のない PgBouncer のような外部接続プーリングツールを使用する場合に特に役立ちます。そのような場合、このフィールドを空白に設定します。

- **Service 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のパラメーターを指定します。Enterprise Manager ディレクトリに保存されているサーバーの場合、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、サーバーを起動する 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依存します。
  - PEM クライアントが Windows マシンで実行されている場合、十分なアクセス権限がある場合、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、マシン名 `PSE1\pgsql-8.0` などを先頭に付けます。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、十分なアクセス権限があれば、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にフルパスと必要なオプションを提供します。サービス制御機能を実行すると、PEM は status/start/stop キーワードをこれに追加します。例

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

`x` は、PostgreSQL データベースサーバーのバージョンです。

**Post Connection SQL** タブを使用して、 SQL クエリーを指定します。

- **Post Connection SQL** フィールドを使用して、サーバー内のデータベースに行われる接続ごとにオートコミットモードで実行する SQL クエリーを書き込みます。

**Tags** タブを使用して、タグを追加します。

- **Tags** タブを使用してカスタムタグを追加します。これは、オブジェクトエクスプローラーツリーのサーバーノードラベルの横に表示されます。

タグを追加するには - **+** ボタンをクリックします。 - **Text** フィールドを使用して、タグ名を指定します。 - **Color** フィールドを使用して、タグのアクセントカラーを選択します。

**Replication** タブを使用して、サーバーが使用するレプリケーションソリューションを選択します。

- サーバーの **レプリケーションソリューション** を使用します。
  - サーバーがレプリケーションソリューション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置き換えることができます。**EFM**
  - PEM でフェールオーバーマネージャーを監視できるようにするには、**EFM cluster name** 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、フェールオーバーマネージャークラスタープロパティファイル名の名前の接頭辞です。たとえば、ファイル名が `efm.properties` の場合、クラスター名は efm.
  - PEM を使用してフェールオーバーマネージャークラスターのステータスをモニタいる場合、**EFM installation path** フィールドを使用して、フェールオーバーの場所を指定しますマネージャーバイナリファイル。デフォルトでは、Failover Manager 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。`x.x` は Failover Manager のバージョンを指定します。 **Patroni**
  - PEM でパトロニクラスターを監視できるようにするには、 **Patroni cluster name** フィールドを使用してクラスター名を指定します。クラスターのデフォルト名は Patroni です。クラスター名は、Patroni 構成ファイルの `Scope` 値で見つけることができます。構成ファイルは YAML 形式であり、多くの場合 patroni.yaml.
  - という名前が付けられます **Patroni インストールパス**を使用して、Patroni バイナリファイルの場所を指定します。一般的な場所は次のとおりです。
    - `/usr/local/bin`
    - `/usr/bin`
  - **Patroni config path** フィールドを使用して、構成ファイルを指定します。一般的な場所は次のとおりです。
    - `/etc/patroni/patroni.yaml`
    - `/etc/patroni.yaml`
    - Patroni プロセスが開始されるディレクトリ **None**
  - レプリケーションソリューションを使用していない場合は、 **None** を選択します。

[ **PEM Agent** ] 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。

[ **Connection Parameters** ] タブで

- **Bound agent** 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マルチプルの Postgres サーバーをモニタできます。

- **Remote Monitoring?** 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はカスタムグラフとダッシュボードで利用できず、リモートサーバーは一部の PEM ユーティリティ Audit Manager、Capacity Manager、Log Manager などからアクセスできません。
- **Host** フィールドに、データベースサーバーに接続するときに使用するエージェントの 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ことができます。
- サーバーへの接続時に使用するエージェントの **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 [ **プロパティ** ] 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は SSL 暗号化が必要です。 |
| prefer | SSL 暗号化が利用できる場合、サーバーとエージェント間で SSL 暗号化を使用します。 |
| allow | サーバーで必要な場合、接続が 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 暗号化を必要とし、認証局によって登録された証明書を使用してサーバーに認証を要求します。 |
| verify-full | SSL 暗号化を必要とし、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サーバーに認証を要求します。 |

SSL 暗号化の使用の詳細については、PostgreSQL documentation を参照してください。

- **Server** 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する場合、このフィールドに EDB Postgres␣
↪Distributed 対応データベース名を指定します。 **Username** フィールドに、サーバーへの
接続時に使用するエージェントのロール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、機能しない機能のリストについては、␣
↪:ref:`Setting agent privileges <Setting agent privileges>` を参照してください。␣
↪Kerberos 認証方法を使用している場合は、このフィールドに pgd_monitor または pgd_
↪superuser ロールを持つユーザーを指定します。
```

  - 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、必要な特権を非ユーザーに付与できます。-スーパーユーザー。 pg_monitor ロールの詳細については、Default Roles を参照してください。 **Username** フィールドに、サーバーへの接続時に使用するエージェントの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、必要な特権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。

- サーバーに接続するときに使用するエージェントのパスワードを **Password** フィールドと **確認パスワード**フィールドに指定します。パスワードを指定しない場合、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。

[ **Advanced** ] タブで

- **Allow takeover?** を **Yes** に設定して、別のエージェントがサーバーを引き継ぎできることを指定します。この機能により、エージェントは、高可用性フェールオーバープロセスの一部としてサーバーが別のホストに移動した場合など、データベースサーバーの監視を引き継ぐことができます。
- 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。 [ **PEM エージェント** ] タブの [ **Connection Parameters** ] 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ーブは、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ーブはデータベース統計を収集できます。

サーバーのプロパティをビューするには、PEM クライアントツリーでサーバー名前を右クリックし、コンテキストメニューから **Properties** を選択します。サーバーのプロパティを変更するには、[ Properties ] 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。

### 7.3.5 接続とバインディングの検証

登録されると、新しいサーバーは PEM ブラウザツリーに追加され、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最初に接続すると、グローバル概要ダッシュボードには、サーバーリストに「不明」のステータスで新しいサーバーが表示される場合があります。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数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。

数分で、Global Overview 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 **Postgres Server Status** リストに含まれます。

5 分後、Global Overview ダッシュボードはまだ新しいサーバーをリストしていない場合は、モニタリングエージェントのログファイルでエラーを確認します。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 > Probe Log Analysis** を選択します。

## 7.4 Managing a PEM server

PEM サーバーの管理に関連するタスクの一部は次のとおりです。

- 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

- Web サーバーの制御

- PEM 認証とセキュリティの管理
- pg_hba.conf ファイルの変更
- プロキシサーバーを使用するように PEM を変更する

### 7.4.1 PEM サーバーとエージェントの起動と停止

PEM サーバーは、それが常駐する Postgres サーバインスタンスが起動、停止、または再起動したときに起動、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 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、対話型のサーバー制御を提供するグラフィカルサービスコントローラが含まれています。Windows Control Panel から Services ユーティリティにアクセスできます。ユーティリティが開いたら、リスト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。

サービスの状態を制御するには、 **Stop** 、 **Pause** 、 **Start** 、または **Restart** を選択します。

Postgres サーバーに接続しているユーザまたはクライアントアプリケーションは、サービスを停止すると突然切断されます。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。

### 7.4.2 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、以下を制御するサービススクリプトの名前。

- EDB Postgres Advanced Server の PEM サーバーは edb-as-<x> または単に edb-as です。
- PostgreSQL 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。

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引き受けて、コマンドを発行します。

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

そこで

service_name はサービスの名前です。

action は、サービスが実行するアクションを指定します。以下を指定します

- start サービスを開始します。
- stop サービスを停止します。
- restart サービスを停止してから開始します。

- status サービスのステータスを確認します。

### 7.4.3 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、対話型のサーバー制御を提供するグラフィカルサービスコントローラが含まれています。以下を制御するサービスの登録名前

- PostgreSQL の PEM サーバーホストは postgresql-<x> です。
- Advanced Server 上の 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、サーバーバージョン番号を示します。

Windows Control Panel で、 Services ユーティリティを開き、Services のリストを表示します。

サービスを停止するには、 **Stop the service** を選択します。サーバーに接続しているユーザまたはクライアントアプリケーションは、サービスを停止すると突然切断されます。

**Pause the service** を選択して、サービスの構成パラメーターをリロードします。**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には、 **Start the service** を選択します。

### 7.4.4 Linux での Web サーバーの制御

Linux では、NGINX と Apache HTTPD の両方がサポートされています。新規インストールのデフォルトは NGINX です。

次のコマンドを実行して、インストールが使用している Web サーバーを確認できます。USE_NGINX=1 を返した場合、インストールは NGINX を使用しています。USE_NGINX=0 を返すか、何も返さない場合、インストールは Apache HTTPD を使用しています。

```
grep USE_NGINX /usr/edb/pem/share/.install-config
```

#### 7.4.4.1 NGINX

次のコマンドで、NGINX サービスのステータスを確認できます。

```
systemctl status nginx
```

サービスが実行されていない場合は、開始できます。

```
systemctl start nginx
```

NGINX が Web サーバーとして使用される場合、PEM は別のサービス edb-uwsgi を使用してアプリケーション自分自身を実行します。

次のコマンドで、edb-uwsgi サービスのステータスを確認できます。

```
systemctl status edb-uwsgi
```

サービスが実行されていない場合は、開始できます。

```
systemctl start edb-uwsgi
```

#### 7.4.4.2 Apache HTTPD

次のコマンドを使用して、Apache サービスのステータスを確認できます。オペレーティングシステムのサービスの名前を<service-name> に置き換えます。これは通常、RHEL 様のシステムの場合は httpd であり、Debian 様のシステムの場合は apache2 です。

```
systemctl status <service-name>
```

サービスが実行されていない場合は、開始できます。

```
systemctl start <service-name>
```

### 7.4.5 Windows での Web サーバーの制御

Windows では、Web サーバーは PEM-HTTPD と呼ばれる Apache HTTPD のバンドルバージョンです。

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。

Status 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 を起動します。

### 7.4.6 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
- アイデント
- LDAP
- RADIUS
- 証明書 SSL
- PAM

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では、次のことを指定します。

- 特定のデータベースに接続できる人
- 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認証接続がフォームを取る安全なパスワード MD5 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ます `pg_hba.conf` ファイル内の一般的なエントリー

```
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があります。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。パスワードが `1safepwd` である postgres 名前付けのユーザーのパスワードファイルのエントリーは、次のフォームます。

```
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` 、または 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。

### 7.4.7 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ーを管理し、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。[ Server Configuration ] ダイアログボックスを開くには、[ **Management > Server Configuration** ] 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。

### 7.4.8 セキュリティの管理

PEM は、Postgres のロール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。

#### 7.4.8.1 ログインロール

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、postgres スーパーユーザーアカウントはサーバーへの最初の接続に使用されますが、セキュリティと監査の両方の目的で、接続する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 インターフェイスの Create – Login/Group Role ダイアログボックス、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、ロールを作成するサーバーのノードを展開します。
2. **Login/Group Roles** 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、 **Create > Login/Group Role** 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、ロールを定義します。

5. 新しいロールの定義が終了したら、[ **Save** ] を選択してロールを作成します。

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 **Delete/Drop** 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。

#### 7.4.8.2 グループロール

グループロールはコンテナとして機能し、データベースの作成などのシステム特権やテーブルへのデータの挿入などのオブジェクト特権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 によるシステムおよびオブジェクト権限の大量管理を簡単にすることです。多くの異なるログインアカウントに個別に特権を割り当てまたは変更するのではなく、単一のロールの特権を割り当てまたは変更し、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。

**Group Roles** ノード PEM ツリーの各登録サーバーの名前の下にありますを使用して、グループロール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は、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へのアクセスを提供します。ロールの作成の詳細については、PostgreSQL documentation を参照してください。

#### 7.4.8.3 PEM の事前定義されたロールを使用して PEM 機能へのアクセスを管理する

Login/Group Role ダイアログボックスを使用して、制限された特権を持つロールに Audit Manager、Capacity Manager、SQL プロファイラーなどの PEM 機能にアクセスを許可できます。PEM の事前定義されたロールにより、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特権を共有します。ユーザーが PEM 拡張機能にアクセスするには、ロールが pem_user ロールと、機能へのアクセスを許可する事前定義されたロールのメンバーである必要があります。**役割**フィールドを使用して、リストから事前定義されたロール名を選択します。

> **注釈**
>
> [**Membership**] タブでロールメンバーシップを追加すると、**WITH INHERIT**チェックボックスがデフォルトで`false` に設定されます。新しいロールが親ロールの特権を継承するには、**WITH INHERIT** を有効にする必要があります。それがないと、ロールはそのメンバーシップに関連付けられている PEM 機能にアクセスできません。

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

この例では、監査マネージャーにアクセスする acctg_clerk という名前のログインロールを作成します。ロールは、任意の時点で、サーバーへの無制限の接続を make ことができます。

PEM の事前定義されたロールを使用して、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。

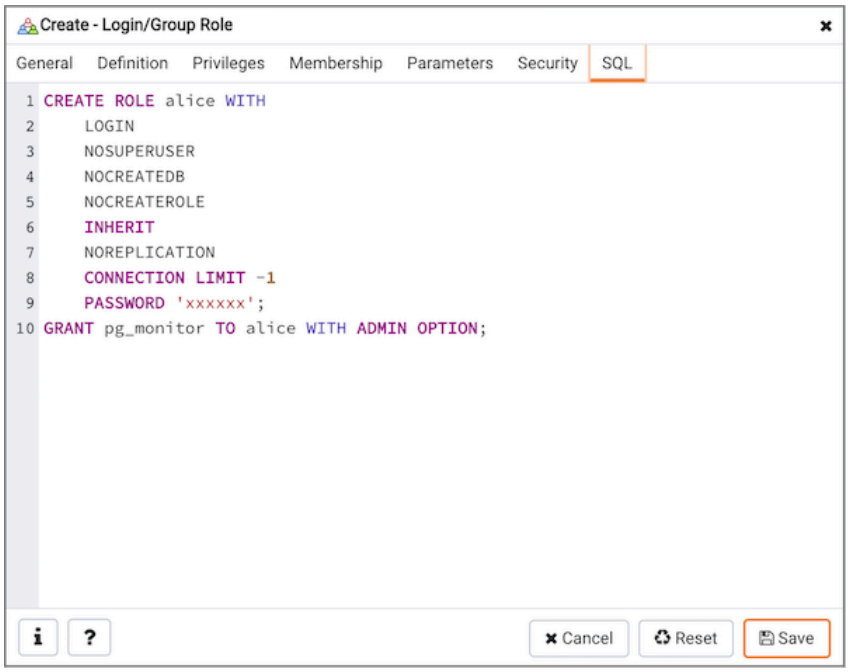



図 1: Create Login Role - SQL tab

| Value | Parent role | Description |
|---|---|---|
| pem_super_admin | | PEM 上のすべてを管理/構成するロール |
| 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/構成のためのロール |
| pem_user | | pem_user ロールを持つユーザ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下の Team フィールドでこのロールが記載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 の構成管理のためのロール |
| 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/実行するロール |
| pem_rest_api | pem_admin | REST API にアクセスするロール |
| pem_server_service_ma | pem_admin | 監視対象データベースサーバーを再起動/リロードできるロールサーバー ID が提供されている場合 |
| pem_manage_schedule_ | pem_admin | スケジュールタスクを構成するロール |
| pem_manage_chart | pem_admin | カスタムチャートを管理/構成するためのロール。 |
| pem_manage_alert | pem_admin | アラートとそのテンプレートを管理/構成するためのロール |
| 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ロール |

次のページに続く

表 8 – 前のページからの続き

| Value | Parent role | Description |
|---|---|---|
| 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ロール |
| pem_config_probe | pem_config, pem_manage_probe | すべての目に見える監視対象オブジェクトのプローブ構成のロール履歴保持、実行頻度、プローブの有効化/無効化 |
| 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_auto_discov | pem_component | データベースサーバーの自動検出ダイアログボックスを実行するロール |
| pem_comp_sqlprofiler | pem_component | SQL プロファイラーを実行するロール |
| pem_manage_efm | pem_admin | フェールオーバーマネージャーを管理するロール |
| pem_comp_capacity_ma | pem_component | Capacity Manager を実行するロール |
| pem_comp_log_manage | pem_component | Log Manager を実行するロール |
| pem_comp_audit_manag | pem_component | Audit Manager を実行するロール |
| pem_comp_bart | pem_component | BART サーバーを構成および管理するロール。 |
| pem_comp_performance | pem_component | パフォーマンス診断を実行するロール。 |
| pem_query_tool | pem_admin | クエリツールにアクセスするロール。 |
| pem_schema_diff_tool | pem_admin | Schema Diff ツールにアクセスするロール。 |
| pem_debugger_tool | pem_admin | デバッガーツールにアクセスするロール。 |
| pem_erd_tool | pem_admin | ERD ツールにアクセスするロール。 |
| pem_grant_wizard | pem_admin | Grant ウィザードにアクセスするロール。 |
| pem_backup_tool | pem_admin | バックアップツールにアクセスするロール。 |
| pem_restore_tool | pem_admin | Restore ツールにアクセスするロール。 |
| pem_import_export_data | pem_admin | データのインポート/エクスポートツールにアクセスするロール。 |
| pem_search_objects_too | pem_admin | Search Objects ツールにアクセスするロール。 |
| pem_maintenance_tool | pem_admin | メンテナンスツールにアクセスするロール。 |
| pem_list_barman | pem_comp_barman, pem_list_all_assets | Barman サーバーをリストするロール。 |
| pem_list_all_assets | pem_super_admin | 'team' フィールドで設定された制限をバイパスして、すべての監視対象リソースとアセットを表示するロール。 |
| pem_manage_assets | pem_admin | データベースサーバー、エージェント、またはツール Barman などを追加または削除するロール。 |

次のページに続く

表 8 – 前のページからの続き

| Value | Parent role | Description |
|---|---|---|
| pem_assign_profile | pem_database_server_registrat<br>pem_manage_profile | サーバーまたはエージェントで監視プロファイルを割り当てまたは割り当て解除するロール。このロールを持つユーザーは、サーバーまたはエージェントの [プロパティ] ダイアログ、[プロファイルの割り当て] ページ、および削除時の再割り当てフローを介して、割り当てられたプロファイルを変更できます。このロールでは、名前、接続設定、ハートビート許容値などの他のサーバーまたはエージェントのプロパティを編集することはできません。 |

#### 7.4.8.4 REST API アクセスに必要なロール

PEM REST API エンドポイントにアクセスするには、ユーザーが `pem_user` および `pem_rest_api` ロールの両方を持っている必要があります。この組み合わせにより、サポートされているすべての API バージョン v1〜v17 のすべてのエンドポイントへの読み取り専用アクセスが付与されます。

書き込み操作 POST、PUT、DELETE には、管理対象のリソースに応じて 1 つ以上の追加管理ロールが必要です。`pem_admin` ロールは、すべての管理ロールを継承し、すべての操作を実行できます。

次の表を、各操作の種類に必要なロールのクイックリファレンスとして使用します。

| Goal | Required roles |
|---|---|
| Read any endpoint | *pem_user* + *pem_rest_api* |
| Manage agents, servers, or server groups | • *pem_manage_assets* |
| Schedule jobs | • *pem_manage_schedule_task* |
| Probe definitions (CRUD, import/export) | • *pem_manage_probe* |
| Probe schedules (copy, configure) | • *pem_config_probe* |
| Alert templates (CRUD, import/export) | • *pem_manage_alert* |
| Alert configuration (copy, configure) | • *pem_config_alert* |
| Charts (import/export) | • *pem_manage_chart* |

次のページに続く

表 9 – 前のページからの続き

| Goal | Required roles |
|---|---|
| Dashboards (import/export) | • *pem_manage_dashboard* |
| Profiles | • *pem_manage_profile* |
| Assign/unassign profiles on an agent or server (without other asset edits) | • *pem_assign_profile* (accepted as alternative to *pem_manage_assets* for the *profile_id* field only) |
| Email groups and server configuration | • *pem_config* |
| Webhooks | • *pem_comp_webhooks* |
| All of the above | *pem_admin* (inherits all management roles) |

#### 7.4.8.5 チームロールを使用する

PEM による監視のためのサーバーを登録するときに、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限定されたロールメンバーの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、指定されたチームの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ロール
- PEM サーバーでスーパーユーザー権限を持つロール

チームロールを作成するには

1. PEM ツリーで、ロールを作成するサーバーのノードを展開します。
2. **Login/Group Roles** 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、 **Create > Login/Group Role** を選択します。
4. [Create - Login/Group Role] 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。

#### 7.4.8.6 オブジェクト権限

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、ロールに十分な特権を付与する必要があります。PEM では、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。

オブジェクトの権限は、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特権を割り当てるには、ツリーでテーブル名前を右クリッ

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。 **Privileges** タブを使用して、テーブルの特権を割り当てます。

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

### 7.4.9 サーバー構成

[ Server Configuration ] ダイアログボックスを使用して、PEM 動作を制御するユーザー設定可能なパラメーターの値を変更できます。[ Server Configuration ] ダイアログボックスにアクセスするには、PEM サーバーに接続し、 **File > Server Configuration** を選択します。

ダイアログの右上隅の検索ボックスにパラメーター名を入力して、リスト内の特定のパラメーターを見つけます。

パラメーター値を変更するには、パラメーター名の右側にある **Value** フィールドに表示される内容を編集します。変更を保存するには、ダイアログボックスの右上隅にある **Save** アイコンを選択します。

### 7.4.10 サーバー構成パラメーター - リファレンス

グローバル構成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場合があります。

| Parameter name | Value and unit | Description |
|---|---|---|
| audit_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、デフォルト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 | ダッシュボードの折れ線グラフと Capacity Manager レポートの箇条書きを有効/無効にします。 |
| cm_data_points_per_rep | 50 | Capacity Manager レポートのチャート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定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/ロールバック分析チャート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 10 – 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の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で Slony レプリケーションのイベント数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_eventlag_timeo | 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 Table Analysis 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ホットテーブル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_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。 |
| 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_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 | 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タイムラグ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_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_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 10 – 前のページからの続き

| Parameter name | Value and unit | Description |
|---|---|---|
| dash_efm_timeout | 300 seconds | ストリーミングレプリケーションダッシュボードでフェールオーバーマネージャーノードの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 Analysis ダッシュボードのチェックポイントチャート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。 |
| 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 Object I/O Details テーブルに表示される行数を指定します。 |
| dash_io_objectio_timeou | 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 Object I/O Details 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チャート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 | サーバーのメモリ分析ダッシュボードのメモリアクティビティ分析チャート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 10 – 前のページからの続き

| Parameter name | Value and unit | Description |
|---|---|---|
| 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円グラフが更新されるまでの秒数を指定します。 |
| dash_objectact_objstora | 15 rows | 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ストレージテーブルに表示する行数を指定します。 |
| dash_objectact_objstora | 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オブジェクトストレージテーブルが更新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トップ 5 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_objtoptal | 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 最大テーブル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 チャート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。 |
| 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チャート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 10 – 前のページからの続き

| Parameter name | Value and unit | Description |
|---|---|---|
| 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チャート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チャート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チャート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_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_patroni_timeout | 300 seconds | ストリーミングレプリケーションダッシュボードのパトロニノードステータスとパトロニクラスターステータスの折れ線グラフの更新間隔を秒単位で指定します。 |
| dash_probe_log_timeout | 300 seconds | プローブログテーブルが更新されるまでの秒数を指定します。 |
| dash_replication_archive | 7 days | Streaming Replication Analysis ダッシュボードの WAL Archive Status チャート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 Archive Status 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 Lag Pages チャート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 Lag Pages 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 10 – 前のページからの続き

| Parameter name | Value and unit | Description |
|---|---|---|
| dash_replication_segmei | 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 ラグセグメントチャートにプロットするデータの日数を指定します。 |
| dash_replication_segmei | 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_timelaş | 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チャートにプロットするデータの日数を指定します。 |
| dash_replication_timelaş | 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_wri | 168 hours | Server Analysis ダッシュボードの Background Writer Statistics 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_buffers_wri | 300 seconds | Server Analysis ダッシュボードで Background Writer Statistics 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_connovervv | 300 seconds | Server Analysis ダッシュボードで Connection Overview 円グラフが更新されるまでの秒数を指定します。 |
| dash_server_database_ti | 300 seconds | Server Analysis ダッシュボードで Databases テーブルが更新されるまでの秒数を指定します。 |
| dash_server_dbsize_spai | 7 days | Server Analysis 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 | Server Analysis 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_timeo | 1800 seconds | Server Analysis ダッシュボードでディス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_server_global_spai | 7 days | Server Analysis ダッシュボードのディスク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_server_sharedbuff_ | 7 days | Server Analysis ダッシュボードの共有バッファ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_sharedbuff_ | 1800 seconds | Server Analysis ダッシュボードで共有バッファー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 | Server Analysis ダッシュボードのテーブルスペースサイズチャート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 10 – 前のページからの続き

| Parameter name | Value and unit | Description |
|---|---|---|
| dash_server_tabspacesiz | 1800 seconds | Server Analysis 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 | Server Analysis ダッシュボードのユーザーアクティビティ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_useract_tim | 1800 seconds | Server Analysis ダッシュボードでユーザーアクティビティ折れ線グラフが更新されるまでの秒数を指定します。 |
| dash_sess_waits_timewa | 300 seconds | セッション待機時間分析ダッシュボードでセッション待機時間ごとの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 | Session Waits Analysis ダッシュボードの Session Waits Details テーブル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の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の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_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の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 | Host Details 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テーブルスペースの概要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 | システム待機数分析ダッシュボードで待機数別の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 | システム待機時間分析ダッシュボードでシステム待機時間ごとの円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 10 – 前のページからの続き

| Parameter name | Value and unit | Description |
|---|---|---|
| dash_sys_waits_waitdtl_ | 300 seconds | システム待機分析ダッシュボードの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。 |
| deleted_objects_data_ret | 3 days | 削除されたオブジェクトたとえば、サーバーまたはエージェントまたは Barman サーバーのプローブデータが pemhistory および pemdata スキーマから削除されるまでの日数を指定します。これらの古いオブジェクトのデータを削除したくない場合、ジョブを削除します。 |
| deleted_probes_retentio | 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を保存する日数を指定します。 |
| down-load_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。 |
| flap-ping_detection_state_cha | 3 | 指定された間隔内で検出された状態変更の数を指定して、特定のアラートをフラッピングとして定義します。フラッピングは、 *N* **を超える状態変更が** *[ N* + 1 *(min(probe_interval) * 2)] 分間発生し、詳細な状態が None 以外の場合に開始されます。 *N* **のデフォルト値は** *2* **または** *3* **で、** *min(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、状態の変更が** *[2 N* * min(probe_interval)] 分間発生したときに終了します。 |
| job_notification_error_li | 1000 | ジョブ失敗通知に含まれるエラー出力の文字制限を指定します。 |
| job_notification_webhoc | | ジョブ失敗通知に使用する Webhook エンドポイントの名前を指定します。ジョブが失敗すると、設定済みの Webhook に通知が送信されます。 |
| 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_transactio | 5 minutes | 長時間実行されていると考えられるまでに、クエリが実行される分数を指定します。 |
| na-gios_cmd_file_name | file_name | パッシブサービスチェック結果を送信する 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。 |
| na-gios_medium_alert_as_ | f | 中レベルの PEM アラートが Nagios で重要であると考えら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にスプールテーブルに保持する日数を指定します。 |

次のページに続く

表 10 – 前のページからの続き

| Parameter name | Value and unit | Description |
|---|---|---|
| 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、リマインダー電子メールを送信する時間を指定します。 |
| send_alert_reminder_ma | true | PEM サーバーがアクティブなアラートのリマインダー電子メールを繰り返し送信するかどうかを指定します。 `false`に設定されている場合、ユーザーは最初のアラート通知のみを受信します。 |
| 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 | `true`の場合、 **Data** タブが各グラフに追加されます。 **Data** 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 での認証を有効/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にするか無効にするかを指定します。 |
| 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 サーバーホストアドレスを指定します。 |
| 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にスプールテーブルに保持す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_authentication_pa | | snmp_security_name で指定された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_authentication_pr | 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_community | public | トラップ送信時に使用される 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_context_engine_ic | | コンテキストエンジン ID、SNMP トラップ送信時の MIB オブジェクトの識別子を指定します。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。 |

次のページに続く

表 10 – 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に 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_enabled | true | SNMP トラップの送信を有効にするか無効にするかを指定します。 |
| 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_privacy_password | | snmp_security_name で指定された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。 |
| 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_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。その可能な値は次のとおりです。- AUTH_NOPRIV - 認証、プライバシーなし。その場合、 NONE オプションはプライバシープロトコルでのみ可能です。AUTH_PRIV - 認証、プライバシー。次に、認証とプライバシープロトコルで NONE オプション選択を許可しないでください。 NOAUTH_NOPRIV - 認証なし、プライバシーなし。その場合、認証とプライバシープロトコルでは NONE オプションのみが可能です。 SNMPv3 でのみ使用されます。 |
| 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_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にスプールテーブルに保持する日数を指定します。 |
| time-zone_for_system_jobs | current server timezone | システムジョブの次回の実行時間をスケジュールするときに使用するタイムゾーンを指定します。 |
| 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。 |

## 7.5 Managing Postgres servers

### 7.5.1 監視対象データベースサーバーの起動と停止

PEM では、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シャットダウンできます。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、[ Server Registration ] ダイアログボックスに値を入力します。ダイアログボックスを使用して、データベースサーバーを PEM エージェントに登録します。これは、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るローカルエージェントです。次のこともできます。

- [ Properties ダイアログボックスで **Store on PEM Server** ] オプションを指定します。
- [ **Advanced** ] タブの **Service ID** 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>` .
  - PostgreSQL の場合、サービス名は `postgresql-<x>` です。 。

<x> は、サーバー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前を選択し、 **Tools > Queue Server Startup** または **Tools > Queue Server Shutdown** を選択することにより、サーバーをスタートまたは停止できます。

### 7.5.2 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、監視対象への PEM のホストの 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。

`pg_hba.conf` ファイルを手動で変更して、PEM サーバーと他のコンポーネント間の接続を許可する必要がある場合があります。たとえば、PEM-HTTPD インストールが PEM サーバーと同じホストにない場合、PEM サーバーホスト上の `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 data 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
```

`pg_hba.conf` ファイルは、選択したエディターで変更できます。ファイルを変更した後、サーバーをリロードして変更を有効にします。

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` エントリーを示しています。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

```

(次のページに続く)

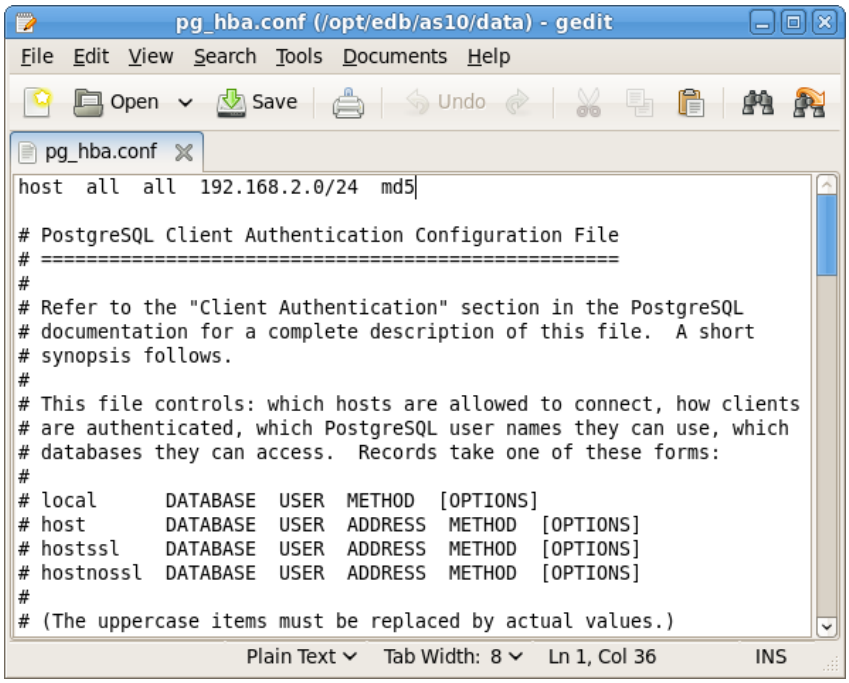



図 2: PEM entries in the pg_hba.conf file

(前のページからの続き)

```
#  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レンジを指定できます。たとえば、アドレス 192.168.10.23、192.168.10.76、および 192.168.10.184 とサーバーからの接続を許可したい場合は、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するように `192.168.10.0/24` の CIDR-ADDRESS を入力します。

```
#  TYPE     DATABASE        USER        CIDR-ADDRESS       METHOD

#  IPv4 local connections:

  host      postgres         all       192.168.10.0/24       md5
```

`pg_hba.conf` ファイルエントリー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保証必要があります。データベースサーバーへの接続時にエラーを受け取った場合は、`pg_hba.conf` ファイルを変更し、接続を許可するエントリーを追加します。

### 7.5.3 データベースとオブジェクトの作成と保守

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の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、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。 [ New Table ] 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
- ファンクション
- トリガー
- ビュー
- コンストレイン

各オブジェクトタイプはツリーにリストされます。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 [ **Create** ] 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、名前付けノードのコンテキストメニューから **Properties** を選択します。

### 7.5.4 テンプレート Linux サービススクリプト

データベースサーバーのサービススクリプトを使用すると、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になる場合があります。サービススクリプトはプラットフォーム固有です。

PEM サーバーが存在する 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のような別のソースからの Postgres サーバーを使用している場合、サービススクリプトを提供する必要があります。

> **注釈**
>
> Rocky Linux または RHEL 8.x では、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。

## 7.6 Managing a PEM agent

Postgres Enterprise Manager ブラウザベースのコンソールを使用して PEM エージェントを管理します。

PEM エージェントはタスクを実行し、エージェントホストおよび監視対象の Postgres インスタンスから PEM サーバーに統計を報告します。単一の PEM エージェントは、エージェントがインストールされている同じホストまたは複数のリモートホストに存在する Postgres の複数のインストール済みインスタンスを監視できます。また、サーバーのシャットダウン、SQL プロファイラーのトレース、カスタムジョブなど、スケジュールした他のタスクを実行する責任もあります。

PEM エージェントを管理するには、次のことができます。

- *Managing job notifications*
- *Managing scheduled jobs*
- *Modifying agent configuration*
- *Reviewing or modifying agent properties*
- *Setting agent privileges*
- *Running multiple PEM agent services on a single server*

### 7.6.1 Managing job notifications

PEM コンソールでは、システム生成ジョブの成功または失敗に応じて SMTP トラップを送信する設定を構成できます。これらのジョブは、スケジュールされたタスクの下にリストされるか、カスタムエージェントジョブです。ジョブ、エージェント、または PEM サーバーレベルデフォルトでこれらのメール通知設定を構成して、指定したユーザーグループにメール通知を送信できます。優先順位はジョブ、エージェント、PEM サーバーの順です。

#### 7.6.1.1 ジョブレベルでジョブ通知を設定する

カスタムエージェントジョブの場合にのみ、ジョブレベルで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、Create Agent Job ウィザードの **Notification** タブで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、ジョブ自分自身の作成中に、`CreateAgent Job` ウィザードの [`Notification` ] タブでメール通知設定を構成できます。

[ **通知** ] タブで

- **Send the notification** フィールドで、メール通知を送信する時期を指定します。
- **Send the notifications** フィールドで、メール通知の送信先のメールグループを指定します。

#### 7.6.1.2 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右クリックし、 **プロパティ** を選択します。[ Properties ダイアログボックスで、 **Job Notifications** タブを選択します。

- 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オーバーライドする場合は、**Overridedefaultconfiguration?** を **Yes** に設定します。残りの設定が有効になります。
- ジョブが正常に完了したときに通知を送信するには、 **Send the notifications** を **Yes** に設定します。
- ジョブが失敗したときにジョブ通知を送信するには、 **Send the notifications** を **Yes** に設定します。
- **電子メールグループ**フィールドで、ジョブ通知の送信先の電子メールグループを指定します。

#### 7.6.1.3 サーバーレベルでジョブ通知を設定する

PEM サーバーレベルでジョブ通知を構成するには、 **Management > Server Configuration** を選択します。[ Server Configurations ] ダイアログボックスで

- **job_failure_notification** を選択して、ジョブが失敗するたびにメール通知を送信します。
- **job_notification_email_group** パラメーターを使用して、メール通知の送信先の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、システムに非定期的な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。
- 失敗、成功、または中断の各ステータスが変更された後にメール通知を送信するには、**job_status_change_notification** を選択します。

### 7.6.2 Managing scheduled jobs

指定した順序で定義した一連のステップを実行する PEM スケジュールされた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/シェルスクリプトが含まれる場合があります。たとえば、データベースサーバーのバックアップを取るジョブを作成し、毎月実行されるタイミングをスケジュールできます。

PEM スケジュールされた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。ジョブを追加するには、**Jobs** ノードを右クリックし、 **Create Job** を選択します。

[ Create - Agent Job ] ダイアログボックスのタブを使用して、PEM スケジュールされたジョブを構成するステップとスケジュールを定義します。

#### 7.6.2.1 ジョブに関する一般的な情報を提供します

**General** 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。

- **Name** フィールドにジョブの名前を入力します。
- ジョブを有効にするには、 **Enabled** を **Yes** に設定します。
- **Comment** フィールドに、ジョブに関するメモを入力します。

#### 7.6.2.2 ジョブステップを定義する

**Steps** 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。 **Add** +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、ステップを定義する [ Step Definition ] ダイアログボックスを開きます。

- **Name** フィールドにステップの名前を入力します。手順は、名前の英数字オーダー実行されます。
- ジョブの実行時にステップを含めるには、 **Enabled** を **True** に設定します。
- **Kind** スイッチを設定して、SQL コードまたはバッチスクリプトを呼び出すジョブステップを設定します。次に、**Code** タブを使用して、ステップの SQL コードまたはバッチスクリプトを入力します。
- **On error** 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 pem エージェントの動作を指定します。
- **Fail** - このステップの処理中にエラーが発生した場合は、ジョブを停止します。
- **Success**- ステップを正常に完了したものとしてマークし、続行します。-** Ignore** - 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** フィールドに、PEM スケジュールされたジョブを作成するエージェントにバインドされたサーバーを入力します。
- **Server** フィールドに、選択したサーバーに関連付けられたデータベースを入力します。
- **Comment** フィールドに、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提供します。
- ステップが SQL コードを呼び出す場合、**に *1* つ以上の *SQL* ステートメントを提供します** SQL クエリー** フィールド.
- ステップがバッチスクリプトを呼び出す場合、**Code** フィールドにスクリプトを提供します。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` などを指定する限り、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 ファイルのパスを提供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` の `batch_script_user` パラメーターのエントリーを変更する必要がありますファイルを開き、スクリプトを実行するユーザーを指定します。このパラメーターには、非 root ユーザーまたは root を指定できます。ユーザーを指定しない場合、または指定され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\EnterpriseDB\PEM\agent` にあります。

ステップで必要な情報をすべて入力したら、[ **Save** ] を選択します。

#### 7.6.2.3 ジョブスケジュールを定義する

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

[ **Schedules** ] タブで、[ **Add** + ] 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、Schedule Definition ダイアログボックスを開きます。

[ **Schedules 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。

- **Name** フィールドにスケジュールの名前を入力します。
- スケジュール **Yes** を使用するには、 **Enabled** を **Yes** に設定します。
- カレンダーセレクターを使用して、スケジュールの開始日時と終了日時を指定します。
- **Comment** フィールドに、スケジュールに関するコメントを入力します。

**Repeat** タブを選択して、スケジュールが cron スタイルの形式で実行日を定義します。ジョブは、[ **Repeat** ] タブで選択された各日付または時刻要素に実行されます。

各フィールドで、値を選択して、フィールドで選択した値のリストに追加します。

- **Days** ボックスのフィールドを使用して、ジョブを実行日を指定します。
- **Week Days** フィールドを使用して、ジョブを実行日を選択します。
- **Month Days** フィールドを使用を押して、ジョブを実行する数値を選択します。日付に関係なく、月の最終日にジョブを実行するには、**Last Day** 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、ジョブを時と分単位で実行する時間を指定します。

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

**Add** (+) を選択して、例外テーブルに行を追加します。次に

- **Date** 列で、カレンダーセレクターを開き、ジョブを実行したくない日付を選択します。 ***** *を指定して、選択した日の時間にジョブを実行しないことを示します。
- **Time** 列で、時間セレクターを開き、ジョブを実行したくない時間を指定します。 ***** *を指定して、選択した日の任意の時点でジョブを実行しないことを示します。

#### 7.6.2.4 メール通知を設定する

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

- **Send the notification** フィールドで、メール通知を送信する時期を指定します。
- **Send the notifications** フィールドで、メール通知の送信先のメールグループを指定します。

スケジュールの定義が終了したら、[ **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。

**Save** 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、ジョブは、定義されたサーバーの PEM ツリーの **Jobs** ノードの下にリストされます。PEM コンソールの **プロパティ**タブには、選択したジョブの高レベルの概要が表示され、 **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するか、ジョブに関する詳細情報を確認するには、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。

### 7.6.3 Modifying agent configuration

パラメーターを構成して PEM エージェントの動作を制御できます。Linux では、これらのパラメーターは構成ファイルに保存されます。Windows では、これらはレジストリキーとして保存されます。

エージェント構成のほとんどは自動的に管理されます。特に指示されない限り、これらの構成パラメーターを手動で変更しないことをお勧めします。

> **注釈**
>
> パラメーターを追加、削除、または変更した場合は、エージェントを再起動して変更を適用します。

#### 7.6.3.1 構成ファイルの内容

Linux システムでは、PEM 構成パラメーターは`/usr/edb/pem/agent/etc` にある `agent.cfg` ファイルに保存されます。

| 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。*Platform Compatibility* を使用している場合、これは IP またはホスト名のコンマ区切りリストです。構文の詳細については、libpq documentation を参照してください。 | 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ポート。 *Platform Compatibility* を使用している場合、これはポートのコンマ区切りリストです。構文の詳細については、 libpq documentation を参照してください。 | Port 5432. |
| agent_conn_string | エージェントが PEM バックエンドへの接続に使用する接続文字列。 **接続文字列を使用する** を参照してください。 | |
| pem_agent | PEM エージェントに割り当てられる一意の識別子。 | The first agent is '1', the second agent is '2', and so on. |
| agent_ssl_key | PEM エージェントのキーファイルへの完全なパス。 | /root/.pem/agent.key |
| agent_ssl_crt | PEM エージェントの証明書ファイルへの完全なパス。 | /root/.pem/agent.crt |

次のページに続く

表 11 – 前のページからの続き

| Parameter name | Description | Default value |
|---|---|---|
| agent_flag_dir | 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要求についてチェックされ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にファイルの形式で作成されます。 | Not set by default. |
| log_level | PEM ログファイルに書き込むイベントのタイプ `debug2`、*debug1*、*info*、*warning*、または`error`。これらはロギングの詳細度の降順です。 `debug2`は考えられるすべてのログを記録し、 `error`はエラーのみをログに記録します。 | warning |
| log_location | PEM ワーカーログファイルの場所。 | /var/log/pem/worker.log |
| agent_log_location | PEM エージェントログファイルの場所。 | /var/log/pem/agent.log |
| long_wait | 最初の接続試行が失敗した場合に、PEM サーバーに接続しようとする前に PEM エージェントが待機する最大時間秒単位。 | 30 seconds |
| short_wait | キューで次に実行を待機しているプローブを確認するまでに PEM エージェントが待機する最小時間秒単位。 | 10 seconds |
| alert_threads | 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、 アラートスレッドについて 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. |
| enable_smtp | true に設定されている場合、このエージェントは PEM Web アプリケーションで構成されたとおりにメール通知を送信しようとします。 | true for PEM server host, false for all others. |
| enable_snmp | true に設定されている場合、このエージェントは PEM Web アプリケーションで構成されたとおりに SNMP 通知を送信しようとします。 | 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 | 失敗した場合に PEM エージェントが Webhook への呼び出しを再試行する最大回数。 | Default 3. |
| connect_timeout | エージェントが接続を待機する秒単位の最大時間 10 進整数文字列。 | Not set by default. Set to 0 to indicate for the agent to wait indefinitely. |
| query_timeout | エージェントがクエリをキャンセルするまでに実行を許可する最大秒単位の時間。タイムアウトを無効にするには 0 に設定します。このパラメーターは、デフォルトの構成ファイルには含まれません。`agent.cfg`に手動で追加して、デフォルトをオーバーライドします。 | 900 |

次のページに続く

表 11 – 前のページからの続き

| Parameter name | Description | Default value |
|---|---|---|
| allow_server_restart | TRUE に設定されている場合、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/無効になる場合があります。 | False |
| max_connections | 接続スロットラーが使用す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が保持される最小秒数を指定します。 MaxConnections で指定された値に到達し、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 に設定されている場合、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す。 | 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。 TRUE に設定している場合、`max_connections`パラメーターを 1 より大きい値に設定する必要があります。 | false |
| batch_script_dir | スクリプトファイルアラート用が保存されるパス。 | /tmp |
| connection_custom_setup | 監視対象サーバーとの新しい接続が作成されたときに呼び出される SQL コードを提供するために使用されます。 | Not set by default. |
| ca_file | HTTPS サーバー証明書を検証するためにプラットフォームのデフォルトの代わりに使用する CA 証明書へのパス。これは、より具体的な`*_ssl_ca_crt`設定でオーバーライドされない限り、Webhook および PG Backup API エンドポイントに使用され、Barman モニタリングに使用されます。以下を参照してください。 | Not set by default. |
| 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。 | None |
| alert_reconnect_delay | PEM サーバーに接続した後、アラートの評価を開始する前にアラートスレッドが待機する秒単位の時間。これは、サーバーがエージェントからハートビートを受信する機会がある前に評価されるアラートによって発生する誤ったアラームを防ぐのに役立ちます。 | 30 |

次のページに続く

表 11 – 前のページからの続き

| Parameter name | Description | Default value |
|---|---|---|
| **Webhook parameters** | 次のオプションは複数回指定できます。毎回、オプションの前に`[WEBHOOK/<name>]`フォームのヘッダーを配置します。`<name>`は、以前に作成した Webhook の名前です。これらの設定は、pemworker ユーティリティを使用して Webhook が追加されるときに挿入されます。 PEM Web アプリケーションから Webhook を追加する場合、必要な設定を手動で追加する必要があります。 | |
| webhook_ssl_crt | Webhook エンドポイントに接続するときに使用するクライアント証明書への完全なパス。 | |
| webhook_ssl_key | `webhook_ssl_crt`に関連付けられたキーへの完全なパス。 | |
| webhook_ssl_crl | Webhook エンドポイントのサーバー証明書を検証するときに使用する CRL ファイルの完全なパス。 | |
| webhook_ssl_ca_crt | Webhook エンドポイントのサーバー証明書を検証するときに使用する CA 証明書ファイルへの完全なパス。 | |
| allow_insecure_webhooks | true に設定されている場合、エージェントはエンドポイントサーバー証明書を検証できません。このオプションを使用して、PEM がプレーン HTTP エンドポイントおよび信頼できない HTTPS エンドポイントを呼び出すことができます。注意して使用してください。 | false |
| **Barman parameters** | 次のオプションは複数回指定できます。毎回、オプションの前に`[BARMAN/<n>]`フォームのヘッダーを置きます。`<n>`は、以前に登録した Barman サーバーの ID です。これらの設定は、pemworker ユーティリティを使用して Barman が登録追加されるときに挿入されます。 PEM Web アプリケーションから Barman を登録する場合、必要な設定を手動で追加する必要があります。 | |
| ssl_crt | PG Backup API エンドポイントに接続するときに使用するクライアント証明書への完全なパス。 | |
| ssl_key | `ssl_crt`に関連付けられたキーへの完全なパス。 | |
| ssl_ca_crt | PG Backup API エンドポイントのサーバー証明書を検証するときに使用する CA 証明書ファイルへの完全なパス。 | |

#### 7.6.3.2 レジストリの内容

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\EnterpriseDB\PEM\agent
```

レジストリには、表に示されている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ポート。 | Port 5432. |
| AGENT_CONN_STRIN | エージェントが PEM バックエンドへの接続に使用する接続文字列。 接続文字列を使用する を参照してください。 | |
| AgentID | PEM エージェントに割り当てられる一意の識別子。 | 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。 | %APPDATA%\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。 | %APPDATA%\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要求についてチェックされ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にファイルの形式で作成されます。 | Not set by default. |
| LogLevel | PEM ログファイルに書き込むイベントの種類を指定します。これらはロギングの詳細度の降順です。 `debug2`は考えられるすべてのログを記録し、 `error`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 サーバーに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 |
| shortWait | どのプローブが次にキューにある実行待機中かを確認するまでに PEM エージェントが待機する秒単位の最小時間。 | 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、 アラートスレッドについて 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 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. |

次のページに続く

表 12 – 前のページからの続き

| Parameter name | Description | Default value |
|---|---|---|
| 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 | 失敗した場合に PEM エージェントが Webhook へ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秒単位の最大時間 10 進整数文字列。 | Not set by default. If set to 0, the agent waits indefinitely. |
| QueryTimeout | エージェントがクエリをキャンセルするまでに実行を許可する最大秒単位の時間。タイムアウトを無効にするには 0 に設定します。 | 900 |
| AllowServerRestart | TRUE に設定されている場合、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/無効になる場合があります。 | true |
| MaxConnections | 接続スロットラーが使用す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が保持される最小秒数を指定します。 MaxConnections で指定された値に到達し、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 に設定されている場合、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す。 | 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。 | /tmp |
| ConnectionCustomSetup | 監視対象サーバーとの新しい接続が作成されたときに呼び出される SQL コードを提供するために使用されます。 | Not set by default. |
| AllowBatchJobSteps | true に設定されている場合、バッチ/シェルスクリプトが実行されます。スクリプトは、PEM エージェントが実行されているユーザーアカウントによって実行されます。 | None |
| AlertReconnectDelay | 秒単位の時間。アラートスレッドは、PEM サーバーに接続した後、アラートの評価を開始する前にこの時間待機します。これは、サーバーがエージェントからハートビートを受信する機会がある前に評価されるアラートによって発生する誤ったアラームを防ぐのに役立ちます。 | 30 |

次のページに続く

表 12 – 前のページからの続き

| Parameter name | Description | Default value |
|---|---|---|
| **Webhook parameters** | 次のオプションは複数回指定できます。これらはサブキー `SoftwareEnterpriseDBPEMagentWEBHOOK<name>`に配置する必要があります。`<name>`は、以前に作成した Webhook の名前です。これらの設定は、Webhook の作成時に自動的に追加されます。手動で追加することはお勧めしません。 | |
| WebhookSSLCrt | Webhook エンドポイントに接続するときに使用されるクライアント証明書への完全なパス。 | |
| WebhookSSLKey | `webhook_ssl_crt`に関連付けられたキーへの完全なパス。 | |
| WebhookSSLCrl | Webhook エンドポイントのサーバー証明書を検証するときに使用する CRL ファイルの完全なパス。 | |
| WebhookSSLCaCrt | Webhook エンドポイントのサーバー証明書を検証するときに使用する CA 証明書ファイルへの完全なパス。 | |
| AllowInsecureWebhooks | true に設定されている場合、エージェントはエンドポイントサーバー証明書を検証できません。このオプションを使用して、PEM がプレーン HTTP エンドポイントおよび信頼できない HTTPS エンドポイントを呼び出すことができます。注意して使用してください。 | false |
| **Barman parameters** | 次のオプションは複数回指定できます。これらはサブキー `SoftwareEnterpriseDBPEMagentBARMAN<n>`に配置する必要があります。`<n>`は、以前に登録した Barman サーバーの ID です。これらの設定は、Barman サーバーが登録されるときに自動的に追加されます。手動で追加することはお勧めしません。 | |
| SSLCrt | PG Backup API エンドポイントに接続するときに使用されるクライアント証明書への完全なパス。 | |
| SSLKey | `ssl_crt`に関連付けられたキーへの完全なパス。 | |
| SSLCaCrt | PG Backup API エンドポイントのサーバー証明書を検証するときに使用する CA 証明書ファイルへの完全なパス。 | |

#### 7.6.3.3 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たアラートスレッドの数は、 `alert_threads` または `AlertThreads` パラメーターによって決まります。通常、PEM サーバーにあ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で、多くの監視対象サーバー、多くの有効なアラート、または高いアラート

頻度が原因で、アラートが構成された頻度で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、PEM サーバーにあるエージェントでこのパラメーターを約 8 に設定し、他のすべてのエージェントで 0 を設定し、それに応じてチューニングアップまたはダウンすることをお勧めします。

このパラメーターを調整する場合、どのエージェントでも任意のアラートを処理できるため、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は必要ありません。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のアラートスレッドの合計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ているサーバーに関連するアラートのパフォーマンスは追加されません。

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、必要以上に多くのスレッドを割り当てると、PEM サーバーのメモリと CPU の使用量が追加されます。

### 7.6.4 Reviewing or modifying agent properties

PEM Agent Properties ダイアログボックスを使用して、選択した PEM エージェントに関する情報を取得します。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 **Properties** を選択します。次に、PEM エージェントに関する情報を確認または変更できます。

- **Description** フィールドを使用して、PEM クライアントのツリーに表示される PEM エージェントの説明を表示または編集します。
- 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編成できます。**Group** 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、エージェントによって監視され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、接続しているチームメンバーに 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。セキュリティ目的には使用しないでください。
- **Heartbeat Tolerance** フィールドを使用して、ハートビートがどれだけ遅れても見逃したとみなされるかを表示または編集します。**Minutes** または **Seconds** フィールドの横にあるセレクターを使用して、値を変更します。
- **ジョブ通知**タブを使用して、エージェントレベルでメール通知設定を構成します。*Managing job notifications* を参照してください。
- **Agent Configurations** 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ーの現在値が表示されます。 - **Category** 列には、対応するパラメーターのカテゴリー構成または機能が表示されます。

### 7.6.5 Running multiple PEM agent services on a single server

単一のサーバーで複数の PEM エージェントサービスを実行できます。移行中に、データベースサーバーごとに複数の EDB Postgres Enterprise ManagerPEM エージェントを実行するのが標準の方法です。

各エージェントサービスは、独自の構成ファイル、SSL 証明書、およびシステムサービスを使用する別個のプロセスとして実行されます。すべてのエージェントは、同じ PEM サーバーに独立して報告します。

> **注釈**
>
> 単一の PEM エージェントは、同じホスト上の複数の Postgres インスタンスを既にモニタできます。単一のサーバーで複数の PEM エージェントを実行することは、独立した監視コンテキストまたは組織の境界に個別のエージェントが必要な場合にのみ必要です。

#### 7.6.5.1 前提条件

- PEM エージェントパッケージがホストにインストールされています。
- ホストで root または管理者アクセスがある。
- ホストから PEM サーバーにアクセスできます。

#### 7.6.5.2 Linux

##### 各追加エージェントを登録する

各エージェントインスタンスには、独自の構成ディレクトリと SSL 証明書ディレクトリが必要です。`--config-dir` および`--cert-path` を使用して、エージェントごとに個別のディレクトリを指定します。

1. 追加エージェントの構成と証明書用のディレクトリを作成します。

```
mkdir -p /etc/edb/pem/agent2/etc
mkdir -p /root/.pem/agent2
chmod 700 /root/.pem/agent2
```

2. 新しい構成ディレクトリと証明書パスを指定して、追加エージェントを登録します。

```
export PEM_SERVER_PASSWORD=<pem_admin_password>

/usr/edb/pem/agent/bin/pemworker --register-agent \
--pem-server <new-pem-server-ip> \
--pem-port 5432 \
--pem-user enterprisedb \
--display-name "agent-host1-2" \
--config-dir /etc/edb/pem/agent2/etc \
--cert-path /root/.pem/agent2
```

PEM は、一意のエージェント ID を自動的に割り当てます。 `agent.cfg` ファイルと SSL 証明書は、指定したディレクトリに作成されます。

3. 新しい `agent.cfg` のログファイルパスを更新して、デフォルトエージェントとの競合を回避します。

```
vi /etc/edb/pem/agent2/etc/agent.cfg
```

`log_location` および `agent_log_location` を一意のパスに設定します。例

```
log_location=/var/log/pem/agent2-worker.log
agent_log_location=/var/log/pem/agent2.log
```

#### 追加エージェントごとに systemd サービスを作成する

デフォルトの pemagent サービスは、デフォルトの構成ディレクトリを使用してエージェントを実行します。追加エージェントを実行するには、追加エージェントの構成をポイントする新しいサービスユニットファイルを作成します。

1. デフォルトのサービスファイルをコピーします。

```
cp /usr/lib/systemd/system/pemagent.service \
/usr/lib/systemd/system/pemagent2.service
```

2. 新しいサービスファイルを編集し、新しい構成ディレクトリを使用するように ExecStart を更新します。

```
vi /usr/lib/systemd/system/pemagent2.service
```

ExecStart 行を更新して、新しい agent.cfg へのパスを使用して-c フラグを渡します。

```
ExecStart=/usr/edb/pem/agent/bin/pemagent -c /etc/edb/pem/agent2/etc/agent.cfg
```

3. systemd デーモンをリロードして、新しいサービスファイルを選択します。

```
systemctl daemon-reload
```

4. 新しいエージェントサービスを有効にして開始します。

```
systemctl enable --now pemagent2
```

5. サービスが実行されていることを確認します。

```
systemctl status pemagent2
```

同じホストで実行する追加エージェントサービスごとに、これらの手順を繰り返します。それぞれに個別のディレクトリパス、ログファイルパス、およびサービス名を使用しますたとえば、agent3 、pemagent3.service など。

### 7.6.5.3 Windows

単一の Windows サーバーで複数の PEM エージェントサービスを実行することは、サポートされていません。

### 7.6.5.4 エージェント登録の確認

各エージェントサービスを開始した後、すべてのエージェントが PEM Web アプリケーションに登録されアクティブであることを確認します。

1. PEM ブラウザーで、Object Explorer で **PEM Server Directory > PEM Agents** を展開します。

2. 各エージェントが **アクティブ**ステータスで表示されることを確認します。

3. エージェントログファイルに接続または認証エラーがないかを確認します。

verify-registration サブコマンドを使用して、各エージェントを検証することもできます。

```
pemworker --verify-registration --config-dir /etc/edb/pem/agent2/etc
```

終了コードおよび追加オプションの詳細については、*Registering a PEM agent* を参照してください。

### 7.6.6 Setting agent privileges

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。これらの特権により、PEM エージェントは、システムの使用状況について監視対象ホストおよびデータベースサーバーで無制限のプローブを呼び出し、情報を取得して PEM サーバーに返すことができます。

#### 7.6.6.1 root ユーザーと非 root ユーザー

完全な機能を実現するには、root として PEM エージェントを実行します。この表は、特権を制限した場合の影響の高レベルの概要を示しています。

| Feature name | Behavior with non-root user | Works with remote PEM agent |
|---|---|---|
| Audit Manager | The Audit Log Manager might not be able to apply requested modifications if the service can't be restarted. The user running the PEM agent might be different from the user who owns the data directory of the database server. Thus the user running the PEM agent might not be able to change the configuration and also might not be able to restart the services of the database server. | no |
| Capacity Manager | No functionality limitation. | yes Note: There's no correlation between the database server and operating system metrics |

次のページに続く

表 13 – 前のページからの続き

| Feature name | Behavior with non-root user | Works with remote PEM agent |
|---|---|---|
| Log Manager | The Log Manager might not be able to apply requested modifications if the service can't restart. The user running the PEM agent might be different from the user who owns the data directory of the database server. Thus the user running the PEM agent might not be able to change the configuration and also might not be able to restart the services of the database server. | no |
| Manage Alerts | No functionality limitation. | yes Note: When Run Alert Script on the database server is selected, it runs on the machine where the bound PEM agent is running and not on the actual database server machine. |
| Manage Charts | No functionality limitation. | yes |
| Manage Dashboards | Some dashboards might not be able to show complete data. See the オペレーティングシステムの権限の制限によって影響を受ける機能 . | Some dashboards might not be able to show complete data. For example, the operating system information of the database server doesn't appear as not available. |
| Manage Probes | Some PEM probes can't run, and some return incomplete data. See the オペレーティングシステムの権限の制限によって影響を受ける機能 . | Some of the PEM probes don't return information, and some of the functionality might be affected. |
| Scheduled Tasks | Limited. See the オペレーティングシステムの権限の制限によって影響を受ける機能 . | Scheduled tasks work only for the database server. Scripts run on a remote agent. |
| System Reports | No functionality limitation. | yes |
| Core Usage Reports | No functionality limitation. | The Core usage report doesn't show complete information. For example, the platform, number of cores, and total RAM aren't displayed. |

#### 7.6.6.2 オペレーティングシステムの権限の制限によって影響を受ける機能

非 root ユーザーとして PEM エージェントを実行する場合、機能のレベルはエージェントユーザーが持っている権限によって異なります。一部の操作は OS ユーザーの権限の影響を受け、通常の操作には特定の権限が必要です。

#### プローブ

| Probe | Operating system | PEM functionality affected |
|---|---|---|
| Session Information | Linux/Windows | The probe will be missing the following 'per-process' columns if the agent user isn't either root or the same user as Postgres: *memory_usage_mb*, *swap_usage_mb*, *cpu_usage*, *io_read_bytes*, *io_write_bytes*. |
| Patroni Node Status | Linux/Windows | Requires permission to execute *patronictl*. No data is returned otherwise. |
| Patroni Cluster Status | Linux/Windows | Requires permission to execute *patronictl*. No data is returned otherwise. |
| PG HBA Conf | Linux/Windows | Requires permission to read *pg_hba.conf*. No data is returned otherwise. |
| Data and Log File Analysis | Linux/Windows | Requires permission to read PGDATA. No data is returned otherwise. |
| WAL Archive Status | Linux/Windows | Requires read access to the WAL directory. No data is returned otherwise. |
| Failover Manager Node Status | Linux/Windows | Requires permission to execute *efm*. No data is returned otherwise. |
| Failover Manager Cluster Info | Linux/Windows | Requires permission to execute *efm*. No data is returned otherwise. |

#### サービスの再起動

Audit Log Manager と Server Log Manager では、PEM エージェントユーザーが Postgres サービスを再起動して、変更を有効にして機能を動作させる必要があります。エージェントユーザーには、サービスを再起動するための権限が必要です。通常、これにはルートアクセスが必要です。

#### バッチ/シェルタスク

Windows では、エージェントユーザーが管理者権限を持っている場合にのみ PEM エージェントはバッチタスクを実行します。

Linux では、PEM エージェントは、エージェントユーザーが `agent.cfg` で指定された `batch_script_user` になることができる場合にのみシェルタスクを実行できます。これは、root ユーザーと `batch_script_user` に常に当てはまります。

### 7.6.6.3 データベース特権の制限によって影響を受ける機能

PEM エージェントが非スーパーユーザーアカウントを使用して監視対象データベースに接続する場合、使用可能な機能は、そのユーザーに付与された特権に基づいて制限されます。

PEM エージェントは、ほとんどの SQL ベースのプローブの pg_catalog スキーマからデータを読み取ります。通常、pg_monitor ロールをエージェントユーザーに割り当てるだけで十分です。ただし、特定のカタログ機能とプローブは、pg_monitor を超える特権を必要とする場合があります。

また、エージェントユーザーは、プローブを実行する必要があるすべてのターゲットデータベースに接続できる必要があります。エージェントがデータベースに接続できない場合、そのインスタンスではデータベースレベルのプローブは実行されません。このような場合、サーバーレベルのメトリックのみが使用できます。`pg_stat_database` から収集されたものなど。

この表には、Linux および Windows オペレーティングシステムで pg_monitor に加えて権限を必要とするプローブがリストされています。

| Probe | Additional permissions required |
|---|---|
| All PGD probes | *SELECT* permission on tables and views, and *EXECUTE* permission on functions, in the *bdr* schema of the replicated database. |
| Number of WAL Files | *EXECUTE* on *pg_ls_dir()*. |
| Streaming Replication Lag Time | The ability to execute *pg_last_xlog_receive_location()*, *pg_last_xlog_replay_location()*, and *pg_last_xact_replay_timestamp()*. Provided by granting the *pg_wal_monitor* role. |
| Streaming Replication | The ability to execute *pg_xlogfile_name_offset()* and *pg_xlog_location_diff()*. This can be provided by granting the *pg_wal_monitor* role. |
| System Waits & Session Waits | *SELECT* permission on the *system_waits* and *session_waits* views respectively. |
| SQL Protect | *SELECT* on *sqlprotect.edb_sql_protect_stats*. |
| User Information | *SELECT* on *pg_user*. |
| xDB Replication | *SELECT* on EDB Replicator views. |

#### 7.6.6.4 エラー処理

プローブは、十分な権限なしでオペレーティングシステムを検索している場合、プローブは `permission denied` エラーを結果ことができます。プローブが十分な権限なしでデータベースを照会している場合、プローブは `permission denied` エラーを返すか、返されたデータを空の値として PEM チャートまたはグラフに表示する場合があります。

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、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の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、 Event Viewer でログ情報を利用できます。

## 7.7 Scheduled system jobs

PEM には、PEM サーバーで自動的に実行されるビルトインのスケジュールされたシステムジョブのセットが含まれています。これらのジョブは、PEM データベースを維持し、ログテーブルと履歴テーブルから古いデータを削除し、日常的な監視を実行します。*Managing scheduled jobs* とは異なり、システムジョブは PEM によって管理され、ユーザーは変更または削除できません。

システムジョブを表示するには、PEM Web インターフェイスで、メニューから **Management > Scheduled Tasks** を選択します。次に、 **Show system tasks?** トグルを有効にして、カスタムエージェントジョブとともにシステムジョブを表示します。

システムジョブの実行時間をスケジュールするときに PEM が使用するタイムゾーンを構成するには、**Management > Server Configuration** で `timezone_for_system_jobs` パラメーターを設定します。

### 7.7.1 システムジョブリファレンス

| Job name | Description |
|---|---|
| Alert history table cleanup | 定期的に実行して、アラート履歴テーブルから古いデータを削除します。 |
| Audit log table cleanup | 定期的に実行して、監査ログテーブルから古いデータを削除します。 |
| Blackout unreachable servers/agents | 到達できなくなったサーバーとエージェントを黒く表示します。 |
| Check CA certificate expiry | CA 証明書の有効期限を確認します。 |
| Database cleanup | 定期的に実行して、古いプローブ履歴データと古いオブジェクトをデータベースから削除します。 |
| Event history table cleanup | 定期的に実行して、イベント履歴テーブルから古いデータを削除します。 |
| Job log table cleanup | 定期的に実行して、ジョブログテーブルから古いデータを削除します。 |
| Job purge the deleted charts | 定期的に実行して、削除されたカスタムチャートをパージします。 |
| Process alert blackouts | `pem.blackout`レコードに基づいて`alert_blackout`フラグを同期します。 |
| Probe log table cleanup | 定期的に実行して、プローブログテーブルから古いデータを削除します。 |
| Purge deleted custom probes | 定期的に実行して、削除されたカスタムプローブをパージします。 |
| Refresh stale materialized views | フラグテーブルを 1 分ごとにチェックし、マテリアライズドビューが古い場合、または過去 1 時間に更新されていない場合は、`pem.probe_target_view`マテリアライズドビューを更新します。 |
| Server log table cleanup | 定期的に実行して、サーバーログテーブルから古いデータを削除します。 |
| SMTP spool table cleanup | 定期的に実行して、SMTP スプールテーブルから古いデータを削除します。 |
| SNMP spool table cleanup | 定期的に実行して、SNMP スプールテーブルから古いデータをパージします。 |
| Update next run of Check CA certificate expiry Job | Check CA 証明書の有効期限ジョブのスケジュールされた次回の実行時間を更新します。 |

次のページに続く

表 16 – 前のページからの続き

| Job name | Description |
|---|---|
| Update the probe-objects combination | `pem.probe_objects_combo`テーブルにプローブと`parameter_value_list`レコードを挿入または更新します。 |
| Webhook spool history table cleanup | 定期的に実行して、Webhook スプール履歴テーブルから古いデータを削除します。 |
| Webhook spool table cleanup | 定期的に実行して、Webhook スプールテーブルから古いデータを削除します。 |

## 7.8 Managing SSL certificates

PEM は SSL 証明書を使用します。

- ユーザーインターフェイスと REST API を提供する *Web* サーバー証明書への要求を保護するため。
- *PEM* バックエンドデータベースサーバーとエージェントの接続証明書 を保護および認証するため。

### 7.8.1 Web サーバー証明書

PEM は、初期構成中に Web サーバーの SSL 証明書とキーファイルを生成します。証明書は自己署名であるため、ユーザーがブラウザーで PEM Web アプリケーションの URL を開くと、サイトが安全でないことを示す警告が表示されます。

セキュリティを強化し、この警告を削除するには、自己署名 SSL 証明書を、信頼できる認証局によって署名された証明書に置き換えることができます。

#### 7.8.1.1 Web サーバー証明書 NGINX の置き換え

NGINX で独自の SSL 証明書を使用するには、構成ファイルを更新します。

- RHEL 同様のシステムの場合 `/etc/nginx/conf.d/edb-pem.conf`
- Debian 系システムの場合 `/etc/nginx/sites-available/edb-pem.conf`

証明書ファイルと一致するように、構成ファイルのサーバー名とファイルパスを変更します。

```
server {
   # lines omitted here
   server_name  <yourdomain.com>;
   # lines omitted here
}

server {
   # lines omitted here
   server_name  <yourdomain.com>;

   ssl_certificate /path/to/your_domain_name.crt
   ssl_certificate_key /path/to/your_private.key
```

(次のページに続く)

(前のページからの続き)

```
   # lines omitted here
}
```

#### 7.8.1.2 Web サーバー証明書の置き換え Apache HTTPD

Apache HTTPD で独自の SSL 証明書を使用するには、構成ファイル edb-ssl-pem.conf を更新します。証明書ファイルと一致するように、構成ファイルのサーバー名とファイルパスを変更します。

```
ServerName yourdomain.com

#  lines omitted here

SSLCertificateFile /path/to/your_domain_name.crt
SSLCertificateKeyFile /path/to/your_private.key
```

Replacing httpd self-signed SSL certificates を参照してください。

### 7.8.2 PEM バックエンドデータベースサーバーとエージェントの接続証明書

PEM は、PEM エージェントとバックエンドデータベース間に安全な SSL/TLS 接続を実装します。各エージェントには、サーバーとの通信の暗号化と、パスワードの代わりにサーバーでの認証の両方に使用される SSL 証明書があります。

特に構成しない限り、PEM は sslutils 拡張機能を使用して、PEM サーバーが SSL 証明書とキーを生成および署名できるようにします。新しいエージェントが登録されると、PEM サーバーは証明書を発行します。PEM サーバーによって発行された証明書は PEM サーバーによって署名されます。つまり、PEM サーバーが認証局 CA として機能することを意味します。

このアプローチが適切でない場合は、PEM の外部で生成され、信頼できる CA によって署名された SSL 証明書とキーを使用できます。PEM でサポートされているさまざまな SSL モデルの詳細については、*Deciding on an SSL model* を参照してください。既存の PEM インストールを変換する方法については、**信頼できる CA によって署名された証明書とキーを使用する** を参照してください。

#### 7.8.2.1 PEM サーバー上の証明書とキーファイル

PEM サーバーの初期構成中に、次のファイルが PEM サーバーの Postgres データディレクトリに生成されます。

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

ca_certificate.crt および ca_key.key ファイルは、PEM サーバーがエージェント登録中にエージェントに対して生成された証明書に署名するために使用されます。これらは server.crt の署名にも使用されます。手動で置き換えない限り、'ca_certificate.crt' ファイルはルート CA として機能するため、自己署名証明書です。

root.crt ファイルは、ca_certificate.crt ファイルのコピーです。postgresql.conf ファイルの ssl_ca_file パラメーターは、このファイルをポイントしています。

root.crl は、実際の有効期限または割り当てられた有効期限の前に発行 CA によって失効された証明書の証明書失効リスト CRL です。postgresql.conf ファイルの ssl_crl_file パラメーターは、このファイルをポイントしています。

server.crt ファイルは PEM サーバーの署名された証明書であり、server.key ファイルは証明書のプライベートキーです。postgresql.conf ファイルの ssl_cert_file パラメーターは、このファイルをポイントしています。

これらのファイルは、有効期限が近づくと自動的に更新されます。*PEM* **証明書の更新** を参照してください。

#### 7.8.2.2 PEM エージェントの証明書とキーファイル

各エージェントの SSL 証明書とキーは、 *Registering a PEM agent* 中に生成されます。PEM エージェントは、libpq インターフェイスを使用して PEM バックエンドデータベースサーバーに接続し、バックエンドデータベースサーバーのクライアントとして機能します。PEM エージェントは、cert 認証方法を使用し、ssl を有効にしてサーバーに接続します。これは、接続がエージェントのキーを使用して暗号化され、パスワードなどの代わりにエージェントの証明書を使用して認証されることを意味します。

各エージェントには一意の識別子があり、エージェントの証明書とキーには対応する識別子があります。

必要に応じて、エージェントごとに 1 つの証明書を使用するのではなく、すべてのエージェントに同じ証明書を使用できます。詳細については、**共通エージェント証明書とキーペアを生成する** を参照してください。

SSL 証明書を使用して Postgres で接続する方法の詳細については、Postgres ドキュメントの Securing TCP/IP connections with SSL を参照してください。

#### 7.8.2.3 PEM 証明書の更新

SSL 証明書には有効期限があります。PEM によって生成された証明書とキーを使用している場合、PEM は有効期限が前にそれらを置き換えます。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間もなくエキスパイアと、PEM は次のことを行います。

- 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。

**重要**

独自の証明書を提供するか、すべてのエージェントに単一の証明書を使用する場合、自動更新ジョブを無効にします。 PEM サーバーで、次の SQL を実行します。

```
UPDATE pem.job
SET jobenabled=false
WHERE jobname = Check CA certificate expiry;
```

- *Regenerating the agent SSL certificates for PEM using sslutils*

### 7.8.3 共通エージェント証明書とキーペアを生成する

エージェントごとに 1 ユーザーデフォルトではなく、すべての PEM エージェントに単一の Postgres ユーザーを作成して使用することにより、すべてのエージェントに同じ証明書を使用できます。

ユーザーを作成し、エージェント証明書とキーペアを生成し、すべての PEM エージェントにそれらを使用します。

1. PEM バックエンドデータベースに 1 つの共通エージェントユーザーを作成します。ユーザーに pem_agent ロールを付与します。

```
# Running as enterprisedb
psql -p 5444 -U enterprisedb -d pem
CREATE USER pem_agent_common_user;
GRANT pem_agent TO pem_agent_common_user;
```

1. *Generating agent certificates manually* の指示に従って、このユーザーの証明書とキーを生成および構成します。

### 7.8.4 信頼できる CA によって署名された証明書とキーを使用する

PEM SSL 証明書とキーを、信頼できる CA によって署名された証明書とキーに置き換えることができます。

信頼できる CA 証明書とキーを取得した後、**サーバー *SSL* 証明書を、信頼できる *CA* によって署名された証明書に置き換えます。** および **エージェント *SSL* 証明書を、信頼できる *CA* によって署名された証明書に置き換えます。** 証明書とキーを置き換えます。

#### 7.8.4.1 サーバー SSL 証明書を、信頼できる CA によって署名された証明書に置き換えます。

1. 古いサーバー証明書とキーファイルをバックアップします。

```
# Running as root
mkdir /var/lib/edb/as<x>/data/certs
cd /var/lib/edb/as<x>/data/
mv server. *root.*  ca_* /var/lib/edb/as<x>/data/certs
```

1. サーバーのプライベートキーを生成します。

```
openssl genrsa -out server.key 4096
```

1. サーバーの CSR を生成します。

```
openssl req -new -key server.key -out server.csr -subj /C=IN/ST=MH/L=Pune/O=EDB/CN=PEM
```

-subj は、要件に従って提供されます。 CN には、PEM サーバーホストのホスト名またはドメイン修飾フルネームを使用することをお勧めします。

1. 信頼できる CA から CA 証明書 trusted_ca.crt を取得します。
2. CA に CSR に署名し、サーバー証明書を生成するように依頼します。
3. 新しいサーバー証明書の詳細が改ざんされていないこと、指定された詳細と一致することを確認します。

```
openssl x509 -noout -text -in server.crt
```

1. CA から取得した新しい証明書を root.crt ファイルとして使用します。

```
cp trusted_ca.crt root.crt
```

1. 信頼できる CA が CRL を提供しない場合、サーバーによる CRL の使用を無効にします。CRL の使用を無効にするには、postgresql.conf ファイルの ssl_crl_file パラメーターをコメント化します。

**注釈**

以前の CA からの CRL をそのままにして、*ssl_crl_file* をコメントアウトしない場合、サーバーは起動します。ただし、認証は SSL エラーメッセージ `tlsv1 alert unknown ca` で失敗します。エラーは、CRL が原因であることを指定していないため、この問題はコンテキスト外で発生した場合、デバッグが難しい場合があります。

1. 新しい root.crt 、server.key 、および server.crt ファイルをバックエンドデータベースサーバーのデータディレクトリにコピーします。

```
cp root.crt server.key server.crt /var/lib/edb/as<x>/data
```

1. 新しい証明書とキーファイルの所有者と権限をデータディレクトリと同じ名前に変更します。

```
cd /var/lib/edb/as<x>/data/
chown enterprisedb server.* root.crt ca_certificate.crt
chmod 600 server.* root.crt ca_certificate.crt
```

**注釈**

今は PEM サーバーを再起動しないでください。 PEM サーバーを再起動すると、登録されているすべてのエージェントが動作を停止します。

1. 各 PEM エージェントの SSL 証明書を、信頼できる CA 証明書に置き換えます。詳細については、これらの**エージェント *SSL* 証明書を、信頼できる *CA* によって署名された証明書に置き換えます。** を参照してください。

2. PEM バックエンドデータベースサーバーを再起動します。

#### 7.8.4.2 エージェント SSL 証明書を、信頼できる CA によって署名された証明書に置き換えます。

サーバー証明書 server.crt および server.key と CA 証明書 root.crt を置き換えた後にのみ、エージェント SSL 証明書を置き換えます。

1. psql を使用して、SSL 証明書の置き換えに必要なすべてのエージェント識別子 ID を見つけます。

```
psql -U enterprisedb -d pem --no-psqlrc -t -A -c "SELECT id FROM pem.agent WHERE␣
↪active=true"
```

1. 2. キーファイルを必要とするエージェントを特定した後、各エージェントに対して agent<ID>.key を生成します。

```
openssl genrsa -out agent<ID>.key 4096
```

<ID> はエージェント識別子です。

1. 各エージェントの CSR を生成します。

```
openssl req -new -key agent<ID>.key -out agent<ID>.csr -subj /C=IN/ST=MH/L=Pune/O=PEM/
↪CN=agent<ID>
```

-subj は、要件に従って提供されます。 CN の<ID> を適切なエージェント識別子に置き換えます。

**注釈**

エージェントごとに 1 つではなく、すべての PEM エージェントに単一の証明書を使用する場合、共通の Postgres ユーザーを作成し、 *ID* の代わりにこのユーザー名を指定します。 **共通エージェント証明書とキーペアを生成する** を参照してください。

1. CA に CSR に署名し、エージェント証明書を生成するように依頼します。

2. 各ホストに証明書とキーファイルをコピーします。ここで、 <ID> は、/usr/edb/pem/agent/etc/agent.cfg ファイルの agent_id と一致します。

3. 新しい agent<ID>.crt および agent<ID>.key ファイルの所有権と権限を変更します。

```
chown root agent<ID>.crt agent<ID>.key
chmod 600 agent<ID>.crt agent<ID>.key
```

1. 古いエージェント証明書とキーファイルをバックアップします。

```
# Running as root
mkdir root/.pem/certs
mv root/.pem/agent<ID>.* root/.pem/certs
```

1. 各エージェントの証明書とキーファイルを、新しく生成されたファイルに置き換えます。

```
cp agent<ID>.key agent<ID>.crt root/.pem
```

7. PEM エージェントサービスを再起動します。

- Linux の場合

```
# Running as root
systemctl restart pemagent
```

- Windows の場合サービスアプレットを使用して PEM エージェントを再起動します。PEM エージェントサービスの名前は Postgres Enterprise Manager Agent です。[ Services ] ダイアログボックスでサービス名を選択し、[ **Restart the service** ] を選択します。

> **注釈**
>
> 前述のプロセスに従って登録されたエージェントの場合、 Providing an existing SSL certificate and key に示すように、登録時にエージェントに証明書を提供できます。

!!!note 新しいエージェント証明書にパスフレーズが必要な場合、`agent_ssl_passphrase_script` パラメーターを使用してパスフレーズを返すスクリプトへのパスを指定します。パラメーターの詳細については、*Modifying agent configuration* を参照してください。

## 7.8.5 証明書のテスト

認証の問題が発生した場合は、これらのテストを使用して証明書を検証できます。

### 7.8.5.1 ルート証明書と照合して証明書を検証する

サーバーの `root.crt` に従って PEM エージェント証明書が信頼されるかどうかを確認するには、両方の証明書を同じマシンにコピーします。次に、次のコマンドを実行します。

```
openssl verify -verbose -CAfile root.crt  agent1.crt
```

このコマンドは、成功した場合は `agent1.crt:  OK` 、失敗した場合は説明メッセージを返します。

### 7.8.5.2 PEM バックエンドデータベースへのテスト接続を作成する

エージェントユーザーが証明書を使用して接続できるかどうかを確認するには、エージェントが存在するサーバーの root として、次のコマンドを実行します。

```
PGHOST=<pem_host>
PGPORT=<pem_db_port>
PGUSER=agent<ID>
PGSSLCERT=/root/.pem/agent<ID>.crt
PGSSLKEY=/root/.pem/agent<ID>.key
PGSSLMODE=require

export PGHOST PGPORT PGUSER PGSSLCERT PGSSLKEY PGSSLMODE

<psql_path> -A -t -d pem -c "SELECT version()"
```

そこで

- <psql_path> は、 psql 実行可能ファイル、たとえば/usr/edb/as15/bin/psql へのフルパスです。
- <pem_host> は、PEM サーバーのホスト名または IP アドレスです。
- <pem_db_port> は、PEM バックエンドデータベースサーバーポートです。
- <ID> は、テストするエージェントの ID です。/usr/edb/pem/agent/etc/agent.cfg ファイルで定義されています。

**注釈**

**共通エージェント証明書とキーペアを生成する** の指示を使用した場合、 *PGUSER* を共通エージェントのユーザー名に設定する必要があります。

接続が成功すると、データベースサーバーの Postgres バージョンを結果ます。成功は、証明書が有効で、Postgres ユーザーが正しく構成されていることを意味します。

### Regenerating the server SSL certificates for PEM backends using sslutils

PEM バックエンドデータベースサーバー証明書の有効期限が間もなくエキスパイアと、証明書とキーファイルの再生成を計画します。

**重要**

sslutils が使用されている場合、証明書の有効期限が間もなくエキスパイアと PEM はこれらの手順を自動的に実行します。これらの手順は、PEM 証明書とキーを手動で再生成する必要がある場合に、完全を期すために提供されています。

これらの手順は、sslutils を *Deciding on an SSL model* として使用する PEM 展開を対象としています。外部またはローカル CA を使用して証明書を手動で提供した場合、最初に生成したのと同じ方法で証明書を再生成します。

1. 実行中のすべての PEM エージェントを、最初にサーバホストで、次に監視対象ホストで停止します。- Linux の場合

```
# Running as root
systemctl stop pemagent
```

- Windows の場合サービスアプレットを使用して PEM エージェントを停止します。PEM エージェントサービスの名前は Postgres Enterprise Manager Agent です。[ Services ] ダイアログボックスで、サービス名を選択し、[ **Stop the service** ] を選択します。

1. 既存の SSL 証明書とキーをバックアップします。

```
cd /var/lib/edb/as<x>/data
mkdir certs
mv server. *root.*  ca_* certs/
```

1. openssl コマンドを使用して、ca_key.key ファイルを生成します。

```
openssl genrsa -out ca_key.key 4096
```

1. ca_key.key ファイルをバックエンドサーバーのデータディレクトリに移動し、権限を変更します。

```
mv ca_key.key /var/lib/edb/as<x>/data
chmod 600 /var/lib/edb/as<x>/data/ca_key.key
```

1. ca_key.key を使用して ca_certificate.crt ファイルを生成します。

```
openssl req -x509 -nodes -days 3650 -newkey rsa:4096 -keyout ca_key.key -out ca_
↪certificate.crt
```

1. ca_certificate.crt ファイルのアクセス許可を変更します。

```
chmod 600 /var/lib/edb/as<x>/data/ca_certificate.crt
```

1. ca_certificate.crt ファイルを root.crt ファイルとして再利用しファイル。

```
cp /var/lib/edb/as<x>/data/ca_certificate.crt /var/lib/edb/as<x>/data/root.crt
```

1. root.crt ファイルの所有者と権限を変更します。

```
chown enterprisedb /var/lib/edb/as<x>/data/root.crt
chmod 600 /var/lib/edb/as<x>/data/root.crt
```

1. openssl_rsa_generate_crl() ファンクションを使用して、証明書失効リスト root.crl を作成します。

```
psql -U enterprisedb -d pem --no-psqlrc -t -A -c
"SELECT openssl_rsa_generate_crl(/var/lib/edb/as<x>/data/ca_certificate.crt, /var/lib/
↪edb/as<x>/data/ca_key.key)" > /var/lib/edb/as<x>/data/root.crl
```

1. root.crl ファイルの所有権と権限を変更します。

```
chown enterprisedb /var/lib/edb/as<x>/data/root.crl
chmod 600 /var/lib/edb/as<x>/data/root.crl
```

1. openssl コマンドを使用して、server.key ファイルを生成します。

```
openssl genrsa -out server.key 4096
```

1. server.key をバックエンドサーバーのデータディレクトリに移動し、所有権と権限を変更します。

```
mv server.key /var/lib/edb/as<x>/data
chown enterprisedb /var/lib/edb/as<x>/data/server.key
chmod 600 /var/lib/edb/as<x>/data/server.key
```

1. openssl req コマンドを使用して CSR を作成します。

```
openssl req -new -key server.key -out server.csr -subj /C=IN/ST=MH/L=Pune/O=EDB/CN=PEM
```

-subj は、要件に従って提供されます。 CN を PEM サーバーホストのホスト名/ドメイン名として定義します。

1. openssl x509 コマンドを使用して CSR に署名し、サーバー証明書を生成します。server.crt をバックエンドデータベースサーバーのデータディレクトリに移動します。

```
openssl x509 -req -days 365 -in server.csr -CA ca_certificate.crt -CAkey ca_key.key -
↪CAcreateserial -out server.crt
mv server.crt /var/lib/edb/as<x>/data
```

-req は、入力が CSR であることを示します。 -CA および-CAkey オプションは、CSR の署名に使用するルート証明書とプライベートキーを指定します。

1. server.crt ファイルの所有者と権限を変更します。

```
chown enterprisedb /var/lib/edb/as<x>/data/server.crt
chmod 600 /var/lib/edb/as<x>/data/server.crt
```

1. PEM サーバーを再起動します。

```
systemctl restart edb-as-<x>
```

バックエンドデータベースサーバーを再起動すると、PEM サーバーが再起動されます。

1. 各 PEM エージェントの SSL 証明書を再生成します。詳細については、*Regenerating the agent SSL certificates for PEM using sslutils* を参照してください。

### Regenerating the agent SSL certificates for PEM using sslutils

**重要**

sslutils が使用されている場合、証明書の有効期限が間もなくエキスパイアと PEM はこれらの手順を自動的に実行します。これらの手順は、PEM 証明書とキーを手動で再生成する必要がある場合に、完全を期すために提供されています。

これらの手順は、sslutils を *Deciding on an SSL model* として使用する PEM 展開を対象としています。外部またはローカル CA を使用して証明書を手動で提供した場合、最初に生成したのと同じ方法で証明書を再生成します。

- PEM サーバー証明書が再生成された場合。
- PEM エージェント証明書の有効期限が間もなくエキスパイアと。

PEM サーバーと対話するエージェントごとに証明書とキーを再生成し、エージェントにコピーする必要があります。

各エージェントには、pem データベースの pem.agent テーブルに保存されている一意の識別子があります。証明書とキーファイルを、エージェントの識別子に対応する証明書またはキーファイルに置き換える必要があります。

前提条件

- PEM サーバーには証明書があります。
- ca_certificate.crt および ca_key.key は、PEM バックエンドデータベースサーバーのデータディレクトリにあります。
- ca_certificate.crt は root.crt と同じです。
- ca_certificate.crt および ca_key.key は、有効な SSL 証明書とキーです。

PEM エージェント証明書と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それに対応する識別子を見つけます。

```
# Running as enterprisedb
psql -p 5444 -U enterprisedb -d pem --no-psqlrc -t -A -c "SELECT id FROM pem.agent␣
↪WHERE active=true"
```

2. 実行中のすべての PEM エージェントを停止します。

```
# Running as root
systemctl stop pemagent
```

Windows では、サービスアプレットを使用して PEM エージェントを停止します。PEM エージェントサービスの名前は Postgres Enterprise Manager Agent です。 [ Services ] ダイアログボックスで、サービス名を選択し、[ **Stop the service** ] を選択します。

3. 2. キーファイルを必要とするエージェントを特定した後、各エージェントに対して agent.key を生成します。

```
openssl genrsa -out agent<ID>.key 4096
```

ID はエージェント識別子です。

4. 各エージェントの証明書署名要求 CSR を生成します。

```
openssl req -new -key agent<ID>.key -out agent<ID>.csr -subj /C=IN/ST=MH/L=Pune/O=PEM/
↪CN=agent<ID>
```

CN は agent<ID> です。

5. openssl x509 コマンドを使用して CSR に署名し、エージェント証明書を生成します。

```
openssl x509 -req -days 365 -in agent<ID>.csr -CA ca_certificate.crt -CAkey ca_key.
↪key -CAcreateserial -out agent<ID>.crt
```

-req は、入力が CSR であることを示します。 -CA および-CAkey オプションは、CSR の署名に使用するルート証明書とプライベートキーを指定します。

次の証明書とキーファイルのペアを生成する前に、各 PEM エージェントホストで手順 2 および 4 で生成された agent.key および agent.crt ファイルを移動します。

6. 新しい agent<ID>.crt および agent<ID>.key ファイルのアクセス許可を変更します。

```
chmod 600 agent<ID>.crt agent<ID>.key
```

7. 古いエージェント証明書とキーファイルをバックアップします。

```
mkdir root/.pem/certs
mv root/.pem/agent<ID>.* root/.pem/certs
```

8. 各エージェントの証明書とキーファイルを、新しく生成されたファイルに置き換えます。

```
cp agent<ID>.key agent<ID>.crt root/.pem
```

9. PEM エージェントサービスを開始します。

- Linux の場合

```
# Running as root
systemctl start pemagent
```

- Windows の場合サービスアプレットを使用して PEM エージェントを起動します。PEM エージェントサービスの名前は Postgres Enterprise Manager Agent です。[ Services ] ダイアログボックスで、サービス名を選択し、[ **Start the service** ] を選択します。

### Generating agent certificates manually

このページでは、エージェント証明書を手動で生成する方法について説明します。これは、sslutils ではなくローカル CA を使用している場合、または sslutils を使用しているが、共通エージェント証明書の使用に移行する場合に必要です。

これらの手順は、証明書署名要求 CSR を提供する必要がある組織または外部 CA を使用している場合にも役立ちます。この場合、手順 3 および 4 を CA の適切な手順に置き換える必要があります。

これらのさまざまな SSL モデルの詳細については、*Deciding on an SSL model* を参照してください。

### エージェント証明書とキーの生成

1. エージェントが実行されるホストで、エージェントキーを生成します。

```
# Running as root
openssl genrsa -out agent.key 4096
```

1. エージェントの CSR を生成します。

```
openssl req -new -key agent.key -out agent.csr -subj /C=<country>/ST=<state>/L=
↪<locality>/O=<organization>/CN=<agent_user>
```

-subj は組織に合わせて適切に入力し、 <agent_user> は、エージェントが PEM データベースに接続するために使用する Postgres のユーザー名に置き換えます。

1. CA が存在するホストに CSR をコピーします。
2. CA ホストで、openssl x509 コマンドを使用して CSR に署名し、エージェント証明書を生成します。

```
openssl x509 -req -days 365 -in agent.csr -CA ca_certificate.crt -CAkey ca_key.key -
↪CAcreateserial -out agent.crt
```

1. agent.crt をエージェントホストにコピーします。
2. エージェントホストで、agent.crt および agent.key ファイルの権限を変更します。

```
chmod 600 agent.crt agent.key
```

### 新しい証明書とキーを使用するようにエージェントを構成する

次の手順は、この証明書とキーを新しいエージェントが使用するか、既に登録されているエージェントが使用するかによって異なります。

1. 新しい証明書とキーファイルへの絶対パスを提供して、エージェントを登録します。<agent_user> の値は、証明書の一般名と一致し、PEM バックエンドデータベースで pem_agent ロールを持つ Postgres ユーザーである必要があります。

```
export PEM_SERVER_PASSWORD=edb

/usr/edb/pem/agent/bin/pemworker --register-agent \
--pem-server 192.168.99.130 \
--pem-user enterprisedb \
--pem-port 5444 \
--pem-agent-user <agent_user> \
-o agent_ssl_crt=/root/.pem/agent.crt \
-o agent_ssl_key=/root/.pem/agent.key
```

1. pemagent サービスを有効にして開始します。

```
systemctl enable pemagent
systemctl start pemagent
```

1. エージェントホストの agent.cfg ファイルの agent_ssl_key 、および agent_ssl_crt パラメーターを編集して、それぞれ新しいキーと証明書への絶対パスを提供します。エージェントユーザーも変更する場合は、agent_user も編集します。証明書の一般名と一致し、PEM バックエンドデータベースで pem_agent ロールを持つ Postgres ユーザーである必要があります。

```
agent_user=<agent_user>
agent_ssl_key=/<new_location>/agent.key
agent_ssl_crt=/<new_location>/agent.crt
```

1. pemagent サービスを再起動します。

```
systemctl restart pemagent
```

## 7.9 Managing configuration settings

PEM Web アプリケーションは、起動時に複数の構成ファイルを読み取ります。これらのファイルは、Linux では/usr/edb/pem/web 、 Windows では C:\ProgramFiles\edb\pem\server\share\web にあります。

- config.py - これはメイン構成ファイルです。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ことを目的としています。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ナル。パッケージソフトは、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または標準を満たすように調整する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ナル。必要に応じて、config.py と同じディレクトリに作成します。

## 7.10 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。

### 7.10.1 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ている場合、復元することはできません。pemworker ユーティリティを使用してエージェントを再登録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、値が active = f の pem.agent テーブルに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. エージェントのステータスを pem.agent テーブルの true に更新します。

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

x は、ステップ 1 で使用したクエリー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は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。変更が成功すると、サーバーは正常に実行されているように表示されます。

### 7.10.2 構成ファイルパスの更新エージェントのステータスが不明と表示されます

エージェント構成ファイル agent.cfg をデフォルトの場所/usr/edb/pem/agent/etc から別の場所に移動すると、PEM ダッシュボードにエージェントステータスが「不明」と表示される場合があります。その場合、pemagent サービスファイルのエージェント構成ファイルパスの値を更新する必要があります。

1. 次のコマンドを使用して、スーパーユーザーとして pemagent サービスファイルを変更します。

```
# Running as superuser
sudo vi  /usr/lib/systemd/system/pemagent.service
```

2. agent.cfg ファイルパスを更新します。たとえば、Redhat ホストでは、ファイルパスを更新します。

```
ExecStart=/usr/edb/pem/agent/bin/pemagent -c /usr/edb/pem/agent/etc/agent.cfg
```

3. systemd をリロードして、変更したサービススクリプトを更新します。

```
sudo systemctl daemon-reload
```

4. PEM エージェントを再起動します。

```
sudo systemctl restart pemagent
```

### 7.10.3 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

PEM Web インターフェイスを使用して、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ことは、このようなエージェントの数が多い場合、困難になる場合があります。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、ダウンまたは不明なエージェントを削除できます。

次のクエリーを使用して、 *N* 時間以上ダウン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> ));
```

次のクエリーを使用して、不明なステータスの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));
```

# 第8章　USING

## 8.1 Accessing the web interface

PEM サーバーとエージェントをインストールした後、PostgreSQL または EDB Postgres Advanced Server インスタンスの監視と管理を開始するように PEM を構成できます。PEM サーバーインストーラーは、PEM Web インターフェイスをインストールします。インターフェイスを使用して、監視対象サーバーにあるオブジェクトに関する情報を確認したり、PEM サーバーによって収集された統計情報を確認したりできます。

サーバーのインストールが完了したら、次の場所に移動して、選択したブラウザーで PEM インターフェイスを開くことができます。

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

`ip_address_of_PEM_host` は、PEM サーバーのホストの IP アドレスを指定します。

Postgres Enterprise Manager Login ウィンドウのフィールドを使用して、PEM サーバーで自分自身を認証します。

- **Username** フィールドに pem データベースユーザーの名前を入力します。最初に接続するユーザー、これは PEM サーバーのインストール時に指定された名前です。
- 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。

資格情報を入力したら、 **Login** を選択して PEM に接続します。

PEM Web インタフェースが開き、 Global Overview ダッシュボードが表示されます。

[ブラウザ] ペインには、各サーバーにあるデータベースオブジェクトに関する情報へのアクセスを提供するツリーが表示されます。ツリーが展開され、PEM サーバーが監視するサーバーとオブジェクトの階層ビューが表示されます。

PEM Web インターフェイスを使用してデータベースサーバーを管理またはモニタする前に、そのサーバーを PEM サーバーに登録する必要があります。サーバーを登録するときは、サーバーへの接続を説明し、接続の認証情報を提供し、管理設定オプションでエージェントをバインドします。

サーバーは管理対象または非管理対象にすることができます。

- **管理対象** サーバーは PEM エージェントにバインドされます。PEM エージェントは、バインドされたサーバーを監視し、タスクを実行するか、PEM ダッシュボードに表示する統計を報告します。管理対象サーバーは、カスタムアラートなどの拡張 PEM 機能にアクセスできます。サーバーを登録するときに、必要に応じて PEM が管理対象サーバーを再起動できるようにすることもできます。
- **非管理対象**サーバーは PEM エージェントにバインドされません。非管理対象サーバーでデータベースオブジェクトを作成できますが、拡張 PEM 機能カスタムアラートなどは、非管理対象サーバーではサポートされていません。

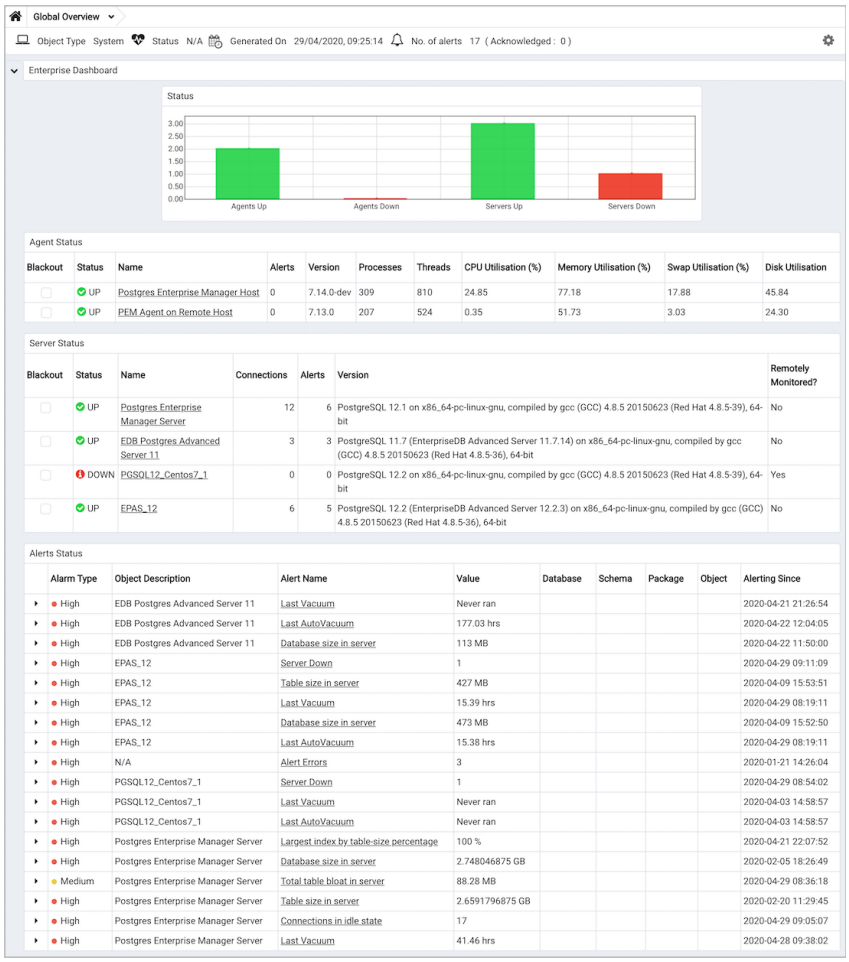



図 1: The Global Overview Dashboard

また、登録するサーバーの pg_hba.conf ファイルで、PEM Web インターフェイスのホストからの接続が許可されていることを確認する必要があります。

### 8.1.1 PEM クライアントオブジェクトブラウザ

Browser ツリーコントロールは、各サーバーに存在するデータベースオブジェクトの情報と管理オプションへのアクセスを提供します。ツリーコントロールが展開され、PEM サーバーによって監視されているサーバーとオブジェクトの階層ビューが表示されます。ロールが必要な特権を保持している場合、ツリーコントロールのノードを右クリックしてアクセスするコンテキストメニューオプションを使用して、新しいオブジェクトを作成し、既存のオブジェクトを変更および削除できます。

ツリーコントロールのノードを展開して、選択したサーバーにあるデータベースオブジェクトの階層ビューを表示します。

- ノードの左側のプラス記号+を使用して、ツリーコントロールのセグメントを展開します。
- ノードの左側にマイナス記号-を使用して、そのノードを閉じます。

コンテキストメニューオプションには、次の 1 つ以上の選択を含めることができます。

| Option | Action |
|---|---|
| **Add named restore point** | 復元ポイントの名前を作成して入力します。 |
| **Backup** | *[ツール] メニュー* ダイアログボックスを開き、データベースオブジェクトをバックアップします。 |
| **Backup Globals** | *[ツール] メニュー* ダイアログボックスを開き、クラスターオブジェクトをバックアップします。 |
| **Backup Server** | *[ツール] メニュー* ダイアログボックスを開き、サーバーをバックアップします。 |
| **Connect Server** | 選択したサーバーとの接続を確立します。 |
| **Create** | 状況依存の選択を提供するコンテキストメニューにアクセスします。選択すると、新しいオブジェクトを作成するための Create ダイアログボックスが開きます。 |
| **CREATE Script** | *Platform Compatibility* を開き、CREATE スクリプトを編集または表示します。 |
| **Dashboards** | PEM ダッシュボードにすばやくアクセスする場合に選択します。 |
| **Delete/Drop** | 現在選択されているオブジェクトをサーバーから削除します。 |
| **Disconnect Database** | データベース接続を終了します。 |
| **Disconnect Server** | 現在選択されているオブジェクトを更新します。 |
| **Drop Cascade** | 現在選択されているオブジェクトとすべての依存オブジェクトをサーバーから削除します。 |
| **Debugging** | Debugger ツールにアクセスします。 |
| **Grant Wizard** | *[ツール] メニュー* ツールにアクセスします。 |
| **Maintenance** | **管理メニュー** ダイアログボックスを開き、VACUUM、ANALYZE、REINDEX、または CLUSTER します。 |
| **Management** | ノードに関連するアクセス管理タスク。 |
| **Properties** | 現在選択されているオブジェクトのプロパティを確認または変更します。 |
| **Refresh** | 現在選択されているオブジェクトを更新します。 |

次のページに続く

表 1 – 前のページからの続き

| Option | Action |
|---|---|
| **Reload Configuration** | サーバーを再起動せずに構成ファイルを更新します。 |
| **Restore** | *[ツール] メニュー* ダイアログボックスにアクセスして、バックアップからデータベースファイルを復元します。 |
| **View Data** | **View Data** オプションを使用して、クエリツールの **データ出力** タブで選択したテーブルに保存されているデータにアクセスします。 |

テーブルおよびネストされたテーブルノードに関連付けられている状況依存メニューは、追加の表示オプションを提供します。

| Option | Action |
|---|---|
| **Import/Export** | Import/Export ダイアログボックスを開き、選択したテーブルにデータをインポートまたはエクスポートします。 |
| **Reset Statistics** | 選択したテーブルの統計をリセットします。 |
| **Scripts** | *Platform Compatibility* を開き、フライアウトメニューから選択したスクリプトを編集または表示します。 |
| **Truncate** | テーブルからすべての行を削除します。 |
| **Truncate Cascade** | テーブルとその子テーブルからすべての行を削除します。 |
| **View First 100 Rows** | 選択したテーブルの最初の 100 行を表示するデータグリッドにアクセスします。 |
| **View Last 100 Rows** | 選択したテーブルの最後の 100 行を表示するデータグリッドにアクセスします。 |
| **View All Rows** | 選択したテーブルのすべての行を表示するデータグリッドにアクセスします。 |
| **View Filtered Rows** | **Data Filter** ポップアップにアクセスして、データセットにフィルターを適用します。 |

### 8.1.2 PEM タブ付きブラウザウィンドウ

PEM クライアントのメインパネルには、ツリーコントロールで現在選択されているオブジェクトに関する情報を表示するタブのコレクションが含まれています。

**Dashboard** タブは状況依存です。サーバーグループまたは PEM エージェントノードから**ダッシュボード**タブに移動すると、 EDB Postgres の Welcome ウィンドウが開きます。ここで次のことができます。

- **Add New Server** アイコンを選択して **データベースサーバーの手動登録** を開き、サーバーへの接続を定義します。
- **Configure PEM** アイコンを選択して *PEM* **サーバー構成の編集** を開き、サーバーパラメーターを変更します。
- **Getting Started** アイコンを選択して新しいタブを開き、EnterpriseDB Web サイトの PEM 入門セクションを表示します。
- **EDB Website** アイコンを選択して、EnterpriseDB Web サイトのホームページに移動します。 EnterpriseDB Web サイトには、今後のイベントやその他のプロジェクトに関するニュースが掲載されています。

- **PostgreSQL Website** アイコンを選択して、PostgreSQL プロジェクトの Web サイトに移動します。PostgreSQL サイトには、最近のリリースおよびその他のプロジェクト情報に関するニュースが掲載されています。
- **EDB Blogs** アイコンを選択して EDB Blog ページに移動します。ここで、Postgres 関連ブログへの最新の従業員の投稿を確認できます。

エージェントまたはサーバーの名前を選択し、**ダッシュボード**タブに移動して、現在選択しているオブジェクトのセッションまたはサーバーアクティビティを確認します。

エージェントまたはサーバーの名前から開くと、 **ダッシュボード**タブは使用統計のグラフィカル分析を提供します。

- サーバーセッションまたはデータベースセッショングラフには、サーバーまたはデータベースとの相互作用が表示されます。
- 1 秒ごとのトランザクショングラフには、サーバーまたはデータベースで発生している 1 秒ごとのコミット、ロールバック、および合計トランザクションが表示されます。
- Tuples In グラフには、サーバーまたはデータベースで挿入、更新、および削除されたタプルの数が表示されます。
- Tuples Out グラフには、サーバーまたはデータベースから取得および返されたタプルの数が表示されます。
- Block I/O グラフには、サーバーまたはデータベースのファイルシステムから読み取られたブロック、またはバッファキャッシュただし、オペレーティングシステムのファイルシステムキャッシュから取得されたブロック数が表示されます。
- Server アクティビティのタブ付きパネルには、セッション情報、セッションロック、準備されたトランザクション、構成を含むテーブルが表示されます。

[ **プロパティ** ] タブに移動して、ツリーコントロールで現在選択されている項目のプロパティを確認します。

[ **SQL** ] タブには、ブラウザツリーコントロールで現在選択されているオブジェクトを生成するために使用される SQL コードが表示されます。

[ **Statistics** ] タブには、ツリーコントロールのオブジェクトごとに収集された統計が表示されます。テーブルに表示される統計は、強調表示されたオブジェクトの種類によって異なります。列見出しを選択して、列に表示されているデータでテーブルを並べ替えます。並べ替え順序を逆にするには、それを再度選択します。次の表は、表示される統計の一部をリストしています。

| Panel | Description |
|---|---|
| PID | 行に関連付けられたプロセス ID。 |
| User | オブジェクトを所有するユーザーの名前。 |
| Database | データベース名。 |
| Backends | データベースへの現在の接続数。 |
| Backend start | バックエンドプロセスの開始時刻。 |
| Xact Committed | 過去 1 週間にデータベースにコミットされたトランザクションの数。 |
| Xact Rolled Back | 過去 1 週間にロールバックされたトランザクションの数。 |
| Blocks Read | 過去 1 週間にメモリから読み取られたブロック数 MB 単位。 |

次のページに続く

表 3 – 前のページからの続き

| Panel | Description |
|---|---|
| Blocks Hit | 先週にキャッシュでヒットしたブロック数 MB 単位。 |
| Tuples Returned | 先週に返されたタプルの数。 |
| Tuples Fetched | 先週に取得されたタプルの数。 |
| Tuples Inserted | 過去 1 週間にデータベースに挿入されたタプルの数。 |
| Tuples Updated | 過去 1 週間にデータベースで更新されたタプルの数。 |
| Tuples Deleted | 過去 1 週間にデータベースから削除されたタプルの数。 |
| Last statistics reset | データベースの最後に統計がリセットされた時間。 |
| Tablespace conflicts | データベース内の削除されたテーブルスペースとのリカバリの競合のためにキャンセルされたクエリの数。 |
| Lock conflicts | データベースのロックとのリカバリの競合のためにキャンセルされたクエリの数。 |
| Snapshot conflicts | データベース内の古いスナップショットとのリカバリの競合のためにキャンセルされたクエリの数。 |
| Bufferpin conflicts | データベース内の固定バッファとのリカバリの競合のためにキャンセルされたクエリの数。 |
| Temporary files | 一時ファイルの合計数。統計コレクターが使用するファイルを含む。 |
| Size of temporary files | 一時ファイルのサイズ。 |
| Deadlocks | データベースのデッドロックとのリカバリの競合のためにキャンセルされたクエリの数。 |
| Block read time | 読み取られたブロックの読み取りに必要なミリ秒数。 |
| Block write time | 読み取られたブロックの書き込みに必要なミリ秒数。 |
| Size | 選択したデータベースのサイズ MB 単位。 |

[ **Dependencies** ] タブには、現在選択されているオブジェクトが依存するオブジェクトが表示されます。データベース構造の整合性を保証するために、サーバーは、他のオブジェクトが依存するオブジェクトを誤ってドロップしないように確認します。`DROP CASCADE` を使用して、別のオブジェクトが依存するオブジェクトを削除する必要があります。

**Dependencies** テーブルには以下が表示されます。

- **Type** フィールド。親オブジェクトタイプを指定します。
- **Name** フィールド。親オブジェクトの識別名を指定します。
- **Restriction** フィールド、現在選択されているオブジェクトと親間の依存関係を説明します。

[ **Dependents** ] タブには、ブラウザツリーで現在選択されているオブジェクトに依存するオブジェクトのテーブルが表示されます。依存オブジェクトは、ブラウザツリーコントロールで現在選択されているオブジェクトに影響を与えることなくドロップできます。

- **Type** フィールドは、依存オブジェクトタイプを指定します。
- **Name** フィールドは、依存オブジェクトの識別名を指定します。
- **Restriction** フィールドは、現在選択されているオブジェクトと親間の依存関係を説明します。

**Monitoring** タブに移動して、*Dashboards* で表示される情報にアクセスします。ダッシュボードには、PEM サーバーによって監視されているオブジェクトに関する統計情報が表示されます。

[ Management ] または [ Tools ] ダイアログボックスから PEM 機能にアクセスすると、PEM は追加のタブを開きます。現在のタブを右クリックし、作業スタイルに合わせて表示をカスタマイズできるコンテキストメニューから選択します。

- 現在選択しているパネルを削除するには、 **Remove Panel** を選択します。
- **Rename Panel** を選択して、現在選択しているパネルの名前を変更します。
- **Detach Panel** を選択して、現在選択しているパネルをデタッチし、使いやすいように再配置します。
- **Add Panel** を選択し、パネルに追加する使用可能なオプションを選択します。

PEM クライアントは、プログラムを終了するときに調整を保存します。PEM クライアントを元の形式にリセットするには、 **File > Reset Layout** を選択します。

### 8.1.3 チャート、グラフ、テーブルコントロールを使用する

PEM クライアントダッシュボードの各グラフィックの右上隅にあるアイコンを使用して、PEM クライアントに表示されるチャート、グラフ、テーブルを制御、ダウンロード、およびカスタマイズします。

**Refresh** アイコン を使用して、PEM プローブから利用可能な最新のコンテンツを表示します。

**Download** アイコン を選択して、チャートまたはグラフの.jpeg または.png 画像をダウンロードします。デフォルトでは、ファイルは.jpeg 形式です。ファイルを.png として保存するには、**Personalize** アイコンを使用してダウンロード形式を変更します。

**フルスクリーン**アイコン を選択して、チャートまたはグラフを展開して PEM クライアントのメインペイン全体に表示されます。

**Personalize** アイコンを選択して、セッションのみにチャートまたはグラフの表示プロパティを変更します。

**Information** アイコンを使用して、チャートまたはグラフに関する情報にアクセスします。

#### 8.1.3.1 グラフィックのパーソナライズ

**Personalize** アイコンを選択すると、[ Personalize chart configuration ] ダイアログボックスが開きます。

[ Personalize chart configuration ] ダイアログボックスのコントロールを使用して、グラフィックのプロパティを変更します。

- **Auto Refresh** コントロールを使用して、更新間の秒数を増減します。
- **Auto Refresh** フィールドを使用して、テーブルまたはチャートに表示されるデータの更新の間の秒数を指定します。
- 必要に応じて、 **Download as** フィールドを使用して、チャートを.jpeg 画像または.png 画像としてダウンロードするかどうかを示します。
- 必要に応じて、 **Colors** セレクターを使用して、チャートで使用する表示色を指定します。
- 必要に応じて、 **Show Acknowledged Alerts** スイッチを **Yes** に設定して、 **Ack'ed** 列のチェックボックスで確認したアラートをテーブルに表示することを示しますフィールドを **No** に設定して、テーブルが確認されたアラートを非表示にすることを示します。確認されたアラートは、アラート定義で指定された時間が経過するまで、テーブルコンテンツから削除されません。

表示プロパティをパーソナライズした後、右上隅のコントロールを使用して変更を適用します。

- **Delete** アイコンを使用して、グラフィックのプロパティをデフォルト設定にリセットします。メニューを使用して、グラフィックのこのインスタンスにのみに変更を適用するか、他のダッシュボードに表示されるときに同じグラフィックに変更を適用するかを指定します。
- **Save** アイコンを使用して、グラフィックのプロパティへの変更を保存します。メニューを使用して、グラフィックのこのインスタンスのみに変更を適用するか、他のダッシュボードに表示されるときは同じグラフィックに変更を適用することを指定します。

### 8.1.4 ブラウザツールバー

ブラウザツールバーには、データの表示やクエリツールなど、頻繁に使用される機能のショートカットが提供されます。このツールバーは [ブラウザ] パネルに表示されます。ボタンは、選択したブラウザノードに基づいて有効/無効になります。

- **Query tool** ボタンを使用して、現在のデータベースコンテキストでクエリツールを開きます。
- **View Data** ボタンを使用して、選択したテーブルに保存されているデータを表示/編集します。
- **Filtered Rows** ボタンを使用して Data Filter ポップアップにアクセスし、表示/編集のためにデータセットにフィルターを適用します。

### 8.1.5 PEM メニューバー

PEM メニューバーは、データベースサーバーとこれらのサーバーに存在するオブジェクトの管理に使用できるコマンドと機能へのアクセスを提供します。オプションが無効になっている場合

- 現在接続しているデータベースサーバは、選択した機能をサポートしていない可能性があります。
- 選択されたメニューオプションは、現在のオブジェクトに適用されない場合があります。
- サーバーへの接続に使用したロールには、選択したオブジェクトを変更するための十分な権限がない可能性があります。

PEM Web インタフェースの上部にあるコンテキスト依存メニューにより、環境をカスタマイズ、PEM のエンタープライズ管理機能へのアクセスを提供できます。

#### 8.1.5.1 ファイルメニュー

**File** メニューを使用して、次のオプションにアクセスします。

| Option | Action |
|---|---|
| Preferences | [ Preferences ] ダイアログボックスを開き、PEM クライアント設定をカスタマイズします。 |
| Lock Layout | サブメニューを開き、UI レイアウトをロックするレベルを選択します。 |
| Server Configuration | [ Server Configuration ] ダイアログボックスを開き、PEM サーバーの構成設定を更新します。 |

次のページに続く

表 4 – 前のページからの続き

| Option | Action |
|---|---|
| Reset Layout | ワークスペースパネルがポップアウトされた場合は、 **レイアウトのリセット** を使用してデフォルトにリセットできます。 |

#### 8.1.5.2 オブジェクトメニュー

**Object** メニューは状況依存です。 **Object** メニューを使用して、次のオプションにアクセスします。

| Option | Action |
|---|---|
| **Create** | 状況依存の選択を提供するメニューにアクセスします。 |
| **Refresh** | 現在選択されているオブジェクトを更新します。 |
| **Delete/Drop** | 現在選択されているオブジェクトをサーバーから削除します。 |
| **Connect Server** | [ Connect to Server ] ダイアログボックスを開き、サーバーとの接続を確立します。 |
| **CREATE Script** | クエリツールを開き、選択したスクリプトを編集または表示します。 |
| **Disconnect Server** | 現在選択されているオブジェクトを更新します。 |
| **BART** | BART 構成の削除、BART バックアップの取得、または BART 構成の再検証のオプションを提供するメニューにアクセスします。 |
| **Clear Saved Password** | データベースサーバーパスワードを保存した場合、保存したパスワードをクリアします。パスワードが保存された後にのみ有効になります。 |
| **Clear SSH Tunnel Password** | ssh トンネルパスワードを保存した場合、保存したパスワードをクリアします。パスワードが保存された後にのみ有効になります。 |
| **Drop Cascade** | 現在選択されているオブジェクトとすべての依存オブジェクトをサーバーから削除します。 |
| **Hide** | 現在選択されているグループを非表示にします。非表示のグループを表示するには、設定で **Show hidden groups** オプションを有効にします。 |
| **Properties** | 現在選択されているオブジェクトのプロパティを確認または変更します。 |
| **Trigger(s)** | 現在選択されているテーブルのトリガーを無効または有効にします。 |
| **Truncate** | テーブルからすべての行を削除する **Truncate** か、テーブルとその子テーブルからすべての行を削除します **Truncate Cascade** 。 |
| **View Data** | データを表示するための複数のオプションを提供するメニューにアクセスします。 |
| **Remove Server** | 選択したサーバーをブラウザツリーから削除します。 |
| **Delete/Drop** | 現在選択されているオブジェクトをサーバーから削除します。 |
| **Connect Database** | 選択したデータベースに接続します。 |
| **Count Rows** | 選択したテーブルの行数をカウントします。 |
| **Reset Statistics** | 選択したテーブルの統計をリセットします。 |
| **Scripts** | 選択したテーブルの CREATE、DELETE、INSERT、SELECT、および UPDATE スクリプト。 |

#### 8.1.5.3 管理メニュー

**Management** メニューを使用して、次の PEM 機能にアクセスします。

| Option | Action |
|---|---|
| **Audit Manager** | Audit Manager を開き、監視対象サーバーで監査を構成します。 |
| **Auto Discovery** | [ Auto Discovery ] ダイアログボックスを開き、監視対象データベースサーバーを見つけてバインドするように PEM エージェントを構成します。 |
| **Capacity Manager** | Capacity Manager ダイアログボックスを開き、履歴またはプロジェクトの将来のリソース使用量を分析します。 |
| **Log Manager** | Log Manager ダイアログボックスを開き、サーバーのログ収集を構成します。 |
| **Alerts** | **Alerts** メニューを開き、 **Manage Alerts** を選択して、アラートの動作を作成または変更します。 |
| **Manage Charts** | **Manage Charts** タブを開き、PEM チャートを作成または変更します。 |
| **Manage Dashboards** | VACUUM、ANALYZE、REINDEX、または CLUSTER への Manage Dashboards ダイアログボックスを開きます。 |
| **Probes** | **Probes** メニューを開き、 **Manage Probes** を選択してプローブを作成または変更します。 |
| **Scheduled Tasks** | [ **Scheduled Tasks** ] タブを開き、保留中または最近完了したタスクを確認します。 |
| **Reports** | [ Reports ] ダイアログボックスを開き、サーバーのアラート履歴レポート、システム構成レポート、またはコア使用状況レポートを生成します。 |
| **Schedule Alert Blackout** | Schedule Alert Blackout ダイアログボックスを開き、サーバーとエージェントのアラートブラックアウトをスケジュールします。 |

#### 8.1.5.4 ダッシュボードメニュー

状況依存の **Dashboards** メニューを使用して、ダッシュボードにアクセスします。

| Option | Action |
|---|---|
| **Alerts** | 選択したノードの Alerts ダッシュボードを開きます。 |
| **Audit Log** | 選択したノードの Audit Log Analysis ダッシュボードを開きます。 |
| **PGD Admin** | 選択したノードの PGD Admin ダッシュボードを開きます。 |
| **PGD Group Monitoring** | 選択したノードの PGD Group Monitoring ダッシュボードを開きます。 |
| **PGD Node Monitoring** | 選択したノードの PGD Node Monitoring ダッシュボードを開きます。 |
| **Database Server** | 選択したノードのデータベース分析ダッシュボードを開きます。 |
| **I/O Analysis** | 選択したノードの I/O Analysis ダッシュボードを開きます。 |
| **Memory** | 選択したノードの Memory Analysis ダッシュボードを開きます。 |
| **Object Activity** | 選択したノードの Object Activity Analysis ダッシュボードを開きます。 |
| **Operating System** | 選択したノードの Operating System Analysis ダッシュボードを開きます。 |

次のページに続く

表 7 – 前のページからの続き

| Option | Action |
|---|---|
| **Probe Log** | 選択したノードの Probe Log Analysis ダッシュボードを開きます。 |
| **Server Log** | 選択したノードの Server Log Analysis ダッシュボードを開きます。 |
| **Session Activity** | 選択したノードの Session activity Analysis ダッシュボードを開きます。 |
| **Storage** | 選択したノードの Storage Analysis ダッシュボードを開きます。 |
| **Streaming Replication** | 選択したノードの Streaming Replication Analysis ダッシュボードを開きます。 |
| **System Wait** | 選択したノードの System Wait Analysis ダッシュボードを開きます。 |
| **Session Waits** | 選択したノードの Session Waits Analysis ダスボードを開きます。 |

#### 8.1.5.5 [ツール] メニュー

**Tools** メニューのオプションを使用して、次の機能にアクセスします。

| Option | Action |
|---|---|
| **Schema Diff** | [ Schema Diff ] ダイアログボックスを開き、2 つのデータベーススキーマ間のスキーマオブジェクトを比較します。 |
| **Search objects** | [ Search Objects ] ダイアログボックスを開き、データベース内のデータベースオブジェクトを検索します。 |
| **Server** | 名前付け復元ポイントの追加、パフォーマンス診断、キューサーバーのスタートアップ、キューサーバーのシャットダウン、クラスタープライマリの置換、EFM クラスターのスイッチオーバー、 SQL プロファイラーなどのさまざまなサーバー関連ツールにアクセスします。 |
| **Query tool** | 現在選択しているオブジェクトのクエリツールを開きます。 |
| **Reload Configuration** | サーバーを再起動せずに構成ファイルを更新します。 |
| **Pause replay of WAL** | WAL ログの再生を一時停止します。 |
| **Resume replay of WAL** | WAL ログの再生を再開します。 |
| **Import/Export** | [ Import/Export Data ] ダイアログボックスを開き、テーブルからデータをインポートまたはエクスポートします。 |
| **Maintenance** | VACUUM、 ANALYZE、REINDEX、または CLUSTER への Maintenance ダイアログボックスを開きます。 |
| **Backup** | [ Backup ] ダイアログボックスを開き、データベースオブジェクトをバックアップします。 |
| **Backup Globals** | クラスターオブジェクトをバックアップするために Backup Globals ダイアログボックスを開きます。 |
| **Backup Server** | サーバーをバックアップするには、 Backup Server ダイアログボックスを開きます。 |
| **Restore** | [ Restore ] ダイアログボックスを開き、バックアップからデータベースファイルを復元します。 |
| **Grant Wizard** | Grant Wizard ツールを開きます。 |
| **Schedule Backup** | BART バックアップの [ Schedule Backup ] ダイアログボックスを開きます。 |
| **New ERD Project(Beta)** | ERD ツールを開き、データベースの設計を開始します。 |

次のページに続く

表 8 – 前のページからの続き

| Option | Action |
|---|---|
| **Storage Manager** | ストレージマネージャーを開き、バックアップファイルをアップロード、削除、またはダウンロードします。 |

#### 8.1.5.6 ヘルプメニュー

**Help** メニューのオプションを使用して、ドキュメントにアクセスするか、PEM インストールに関する情報を確認します。

| Option | Action |
|---|---|
| **Quick Search** | Quick Search フィールドにキーワードを入力します。少なくとも 3 文字を入力すると、メニュー項目の下に一致するすべての可能性が表示され、ヘルプ記事の下に関連ドキュメントが表示されます。メニュー項目の下のオプションを選択して、特定の機能またはオブジェクトのアクションを実行します。ヘルプ記事のいずれかを選択すると、そのトピックのヘルプが開き、別のウィンドウで強調表示されたテキストが表示されます。 **注** メニュー項目の下のオプションが無効になっている場合、情報アイコンを介して情報を提供します。 |
| **REST API Reference** | REST API リファレンスを開きます。 |
| **EDB Website** | ブラウザウィンドウで EDB ウェブサイトを開きます。 |
| **About Postgres Enterprise Manager** | Postgres Enterprise Manager のバージョンとユーザー情報を見つけます。 |

### 8.1.6 チャート、グラフ、テーブルの制御とカスタマイズ

ダッシュボードの各グラフィックの右上隅にあるアイコンを使用して、現在のユーザーセッションの PEM クライアントに表示されるチャート、グラフ、テーブルを制御およびカスタマイズします。

次の表に、アイコンに関する情報を示します。

| Icons | Information |
|---|---|
| | View information about the chart, graph, or table. |
| | Refresh the content of a chart, graph, or table. |
| | Personalize the chart, graph, or table settings for the current user. |
| | Download an image of the chart or graph. |
| | View the legends that are used in the chart, graph, or table. |

次のページに続く

表 10 – 前のページからの続き

| Icons | Information |
| --- | --- |
| | Expand the chart or graph to full-screen. |

## 8.2 Connecting and disconnecting a database server

サーバー接続を定義した後、`Connect to Server` ダイアログを使用してサーバーで認証し、サーバーに保存されているオブジェクトにアクセスします。ダイアログにアクセスするには、PEM クライアントツリーコントロール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 `Connect Server` を選択します。

プロンプトが表示されたら、選択したサーバーの認証情報を入力します。

- `Password` フィールドを使用して、定義されたサーバーに関連付けられたユーザーのパスワードを提供します。
- `Save Password` の横にあるボックスを選択して、今後の接続のためにパスワードを保存するようにサーバーに指示します。パスワードを保存すると、このサーバー定義でデータベースサーバーに再接続するときにプロンプトは表示されません。

サーバーが正常に接続すると、ブラウザーは右下隅の緑色のステータスバーにメッセージを表示します。

接続しようとしたときにエラーメッセージを受け取った場合は、ネットワークが PEM とデータベースサーバーのホストの通信を許可していることを確認します。特定のエラーメッセージの詳細については、Troubleshooting ヘルプページを参照してください。

接続の詳細を確認または変更するには、サーバーの名前を右クリックし、コンテキストメニューから `Properties...` を選択します。

### 8.2.1 サーバーからの切断

サーバーから切断するには、`Browser` ツリーコントロールでサーバー名前を右クリックし、コンテキストメニューから Disconnect Server を選択します。選択したサーバーを切断することを確認するポップアップが表示されます。

## 8.3 Monitoring performance

PEM を使用すると、すべての管理対象サーバーの企業全体のパフォーマンス監視を実装できます。

トップレベルのダッシュボードは Global Overview です。 Global Overview ダッシュボードには、PEM サーバーによって監視されているすべてのサーバーとエージェントのステータスの概要、監視対象サーバーのリスト、および現在トリガーされているアラートの状態が表示されます。

PEM は、他の PostgreSQL 管理ツールにはない多くの利点を提供します。

- 一括デザインの管理 - PEM は、エンタープライズデータベース管理用に設計され、地理的境界を越えた多数のサーバーの管理に取り組むように構築されています。グローバルダッシュボードは、すべてのサーバーの稼働/ダウン/パフォーマンスステータスを視覚的に最新の状態に保ちます。

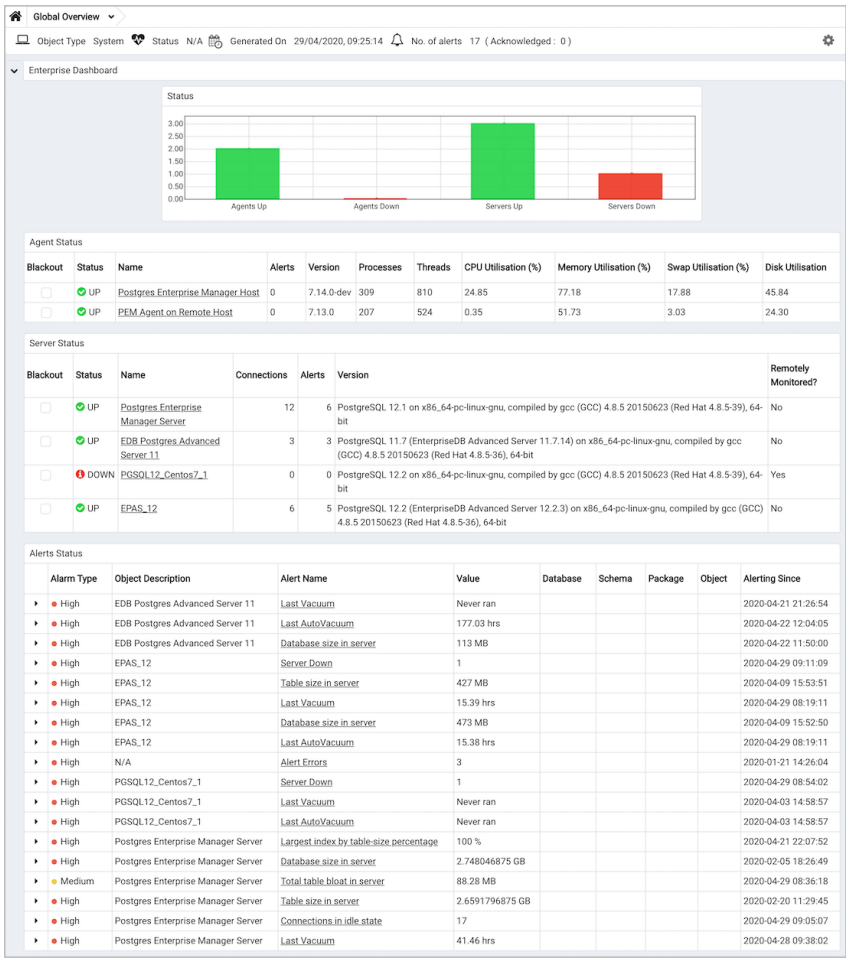



図 2: The Global Overview dashboard

- 分散アーキテクチャ - PEM アーキテクチャは、オペレーティングシステムプラットフォームに関係なく、マシンで統計情報を収集し、リモートで操作を実行する機能を最大化します。
- グラフィカル管理 - データベース管理のすべての側面は、グラフィカルインターフェイスを使用して PEM クライアントで実行できます。サーバーの起動とシャットダウン、構成管理、ストレージとセキュリティ制御、オブジェクト作成、パフォーマンス管理などを単一のコンソールから処理できます。
- フル SQL IDE - PEM には、アドホック SQL クエリー、ストアドプロシージャ/ファンクション開発、およびグラフィカルデバッガーを提供する堅牢な SQL 統合開発環境 IDE が含まれています。
- エンタープライズパフォーマンスモニタリング - PEM は、すべての管理対象データベースサーバーにエンタープライズクラスのパフォーマンスモニタリングを提供します。軽量で効率的なエージェントは、各データベースサーバーの操作と各マシンの基になるオペレーティングシステムのあらゆる側面を監視します。これらは、インターフェイスでパフォーマンスページを簡単に移動するための詳細な統計を提供します。
- プロアクティブなアラート管理 PEM を使用すると、24 時間監視されるメモリやストレージなどの主要メトリックごとにパフォーマンスしきい値を作成できます。しきい値違反は、一元化されたダッシュボードにアラートが送信され、問題の性質と、サーバーの全体的なパフォーマンスを危険にさらす状況を防ぐために必要なアクションを通知します。
- 簡素化されたキャパシティプランニング - 主要なパフォーマンス関連の統計はすべて収集され、指定された期間 PEM リポジトリに保持されます。Capacity Manager ユーティリティを使用すると、さまざまな統計を選択し、傾向分析を実行して、ピーク負荷期間やストレージ消費傾向などを理解できます。ツールの予測メカニズムを使用すると、リソースの使用量を予測できるため、それに応じて計画と予算を立てることができます。
- Audit Manager - Audit Manager は、EDB Postgres Advanced Server インスタンスに監査ログを構成します。データベースへの接続、データベースからの切断、データベースに対して実行された SQL ステートメントなどのアクティビティをログに記録できます。Audit Log ダッシュボードを使用して、ログをフィルタリングおよび表示できます。
- ログマネージャー - ログマネージャーウィザードは、中央のテーブルへのオプショナルのログ収集を使用してサーバーロギングパラメータを構成します。ウィザードを使用して、ログファイルのローテーション、ログの宛先、エラーメッセージの重大度などのロギング動作の設定を指定します。Server Log ダッシュボードを使用して、収集されたサーバーログエントリをフィルタリングおよび確認します。
- SQL ワークロードプロファイリング - PEM には、1 つ以上のサーバーに対して実行される SQL ステートメントをトレースできる SQL プロファイリングユーティリティが含まれています。SQL プロファイリングは、アドホックまたはスケジュールされた方法で実行できます。キャプチャした SQL ステートメントをフィルタリングできるため、実行不良の SQL ステートメントを簡単に特定および調整できます。
- ストリーミングレプリケーションモニタ - ストリーミングレプリケーションダッシュボードをモニタするか、PEM クライアントのオプションを使用して、レプリカノードをプライマリノードにプロモートせることができます。
- セキュアクライアント接続 - PEM は、暗号化された SSH トンネルを介したセキュアクライアント接続をサポートします。フル機能の PEM クライアントには、安全な接続のための接続情報を提供できる SSH トンネル定義ダイアログボックスが含まれています。

- 幅広いプラットフォームのサポート - PEM は、ほとんどのメジャー Linux および Windows プラットフォームをサポートしています。

### 8.3.1 Probes

プローブは、Postgres Enterprise ManagerPEM エージェントによって実行され、パフォーマンスメトリックと監視対象データベースオブジェクトに関する情報を取得するスケジュールされたタスクです。

プローブは次のタイプに分類されます。

- システムプローブ
- *Creating custom probes*

#### 8.3.1.1 システムプローブ

PEM は、5 つのレベルのビルトインプローブを備えています。

- **エージェントレベル** - CPU 使用率、ディスク領域、メモリ使用量、ネットワーク統計などのホスト固有のデータを監視します。
- **サーバーレベル** - バックグラウンドライタ統計、ブロックされたセッション、セッション情報、WAL ファイルを含むグローバルデータベース情報を追跡します。
- **データベース/スキーマレベル** - データベースサイズ、ファンクション統計、テーブル統計、インデックスサイズなどの特定のオブジェクトデータを監視します。
- **ツール固有のプローブ** - Barman バックアップ、フェールオーバーマネージャー HA、Patroni クラスターステータス、およびさまざまなレプリケーション方法ストリーミング、Replication Server のプローブが含まれます。
- **拡張レベル** - *Postgres 分散 PGD プローブ* などの特定の拡張データを監視します。

#### 8.3.1.2 プローブのカスタマイズ

カスタムプローブを作成、変更、コピー、エクスポート/インポート、および削除して、特定のニーズに合わせて監視を調整できます。

**Management > Probes > Manage Probes** タブを使用して、プローブ設定を変更できます。

1. ***Creating custom probes***

カスタムプローブを作成して、3 つの収集方法を使用してビルトインプローブでカバーされない特定のメトリックを監視できます。

- **SQL Query** - カスタム SQL クエリーを作成して、データベースから特定のメトリックを取得します。- **WMI** - WMI を使用して Windows システムからデータを収集します。 - **バッチ** - コマンドラインまたはシェルスクリプトを使用して、システムからデータを収集します。`agent.cfg` ファイルで `allow_batch_probes` を有効にする必要があります。

2. **コア管理タスク**

- **スケジュールの変更** - **デフォルト**スイッチを無効にすることにより、プローブの実行頻度分/秒をカスタマイズします。

- **プローブを有効/無効にする** - ステータスを切り替えます必要に応じて**オン** または **オフ**に設定します。チャートで使用される無効なプローブには、再度有効にするアイコンが表示されます。
- **データ保持** - 自動削除までにプローブ履歴を保存する期間を指定します。
- **構成のコピー** - **Copy probe** ボタンを使用します拡張レベルのプローブを除き、あるオブジェクトから別のオブジェクトにプローブ設定を複製します。

3. **エクスポート/インポートと削除**

- **Export/Import** - **Export / Import** ボタンを使用して、異なる PEM インストール間でプローブ構成を転送します。- **削除** - カスタムプローブは、 **Delete** ボタンを使用して削除できます。システムプローブは削除できません。

### System probes

システムプローブは、PEM によって事前定義された監視プローブです。これらのプローブの定義は変更できません。プローブとは異なる動作が必要な場合は、*Creating custom probes* 。

> **注釈**
>
> PEM 10.4.1 の時点で、UI はシステムプローブのコードの編集を無効にしませんが、変更を保存しようとすると拒否されます。 PEM UI の過去および将来のバージョンは、システムプローブの編集が無効であることを正しく示します。

特に断りのない限り、Postgres Enterprise ManagerPEM は、サーバー、データベース、スキーマ、拡張機能、またはエージェントレベルで次のプローブを有効にします。

| Name | Information monitored by probe | Level |
|---|---|---|
| Background Writer Statistics | バックグラウンドライタに関する情報を監視します。この情報には、次の数が含まれます。時限チェックポイント 要求されたチェックポイント 書き込まれたバッファチェックポイントによって 書き込まれたバッファバックグラウンドライタによって バックグラウンドライタサイクル 書き込まれたバックグラウンドバッファ 割り当てられたバッファ | Server |
| Barman Configuration | Barman ツールのグローバル構成に関する情報。 | Agent |
| Barman Information | Barman ツールに関する情報。 | Agent |
| Barman Server | Barman によって監視される各データベースサーバー構成に関する情報。 | Agent |
| Barman Server Status | Barman によって監視される各データベースサーバーのステータスに関する情報。 | Agent |
| Barman Server Backup | 各データベースサーバーのバックアップに関する情報。 | Agent |
| Barman Server WAL Status | Barman サーバーの WAL ファイルに関する情報。 | Agent |

次のページに続く

表 11 – 前のページからの続き

| Name | Information monitored by probe | Level |
|---|---|---|
| Blocked Session Information | ブロックされたセッションに関する情報。 | Server |
| CPU Usage | CPU 使用率情報。 | Agent |
| Data and Log File Analysis | ログファイルに関する情報。情報には次のものが含まれます。ログファイルの名前 ログファイルが存在するディレクトリ | Server |
| Database Frozen XID | 各データベースの凍結された XID。 | Server |
| Database Size | 監視対象データベースのサイズに関する情報。情報には次のものが含まれます。情報が収集された時刻 データベース名 データベースサイズ MB 単位。 | Server |
| Database Statistics | データベースの統計。この情報には以下が含まれます。バックエンドの数 コミットされたトランザクションの数 ロールバックしたトランザクションの数 読み取られたブロックの数 ブロックヒット数 返された行数 フェッチされた行数 挿入された行数 更新された行数削除された行数 | Server |
| Disk Busy Info | ディスクアクティビティに関する情報。注このプローブは、Mac OS X、Solaris、または HP-UX ではサポートされていません。 | Agent |
| Disk Space | ディスク領域の使用状況に関する情報。この情報には以下が含まれます。使用されたディスク領域の量 使用可能なディスク領域の量 | Agent |
| EDB Audit Configuration | EDB Postgres Advanced Server の監査ログ構成 。 | Server |
| Function Statistics | データベースを監視し、機能に関する情報を取得します。情報には次のものが含まれます。ファンクション名 引数の型 戻り値 | Database |
| Index Size | データベースを監視し、インデックスに関する情報を取得します。情報には次のものが含まれます。インデックスの名前 データが収集された時刻 インデックスのサイズ MB 単位 | Database |
| Index Statistics | インデックス統計。この情報には以下が含まれます。インデックススキャンの数 読み取られた行数 フェッチされた行数 読み取られたブロック数 ヒットしたブロック数 | Database |
| Installed Packages | 現在インストールされているパッケージ。収集される情報には以下が含まれます。インストールされたパッケージの名前 インストールされたパッケージのバージョン プローブが実行された日時 | Agent |

次のページに続く

表 11 – 前のページからの続き

| Name | Information monitored by probe | Level |
|---|---|---|
| IO Analysis | ディスク I/O 情報。この情報には以下が含まれます。読み取られたブロック数 書き込まれたブロック数 読み取り操作の数 書き込み操作の数 プローブが実行された日時 注このプローブは、Mac OS X ではサポートされていません。 | Agent |
| Load Average | CPU 負荷平均。情報には次のものが含まれます。1 分間のロード平均 5 分間のロード平均 15 分間のロード平均 注 このプローブは Windows でサポートされていません。 | Agent |
| Lock Information | ロック情報。情報には次のものが含まれます。データベース名 ロックタイプ ロックモード ロックを保持しているプロセス | Server |
| Memory Usage | システムメモリの使用量に関する情報。情報には次のものが含まれます。合計 RAM MB 単位 空き RAM MB 単位 合計スワップメモリ MB 単位 の空きスワップメモリ MB 共有システムメモリ MB 単位 - 非 Windows システムでは、*shmmax* **値であり、**/proc/sys/kernel/shmmax`から読み取られます。 - Windows では、合計メモリと同じです。 | Agent |
| Network Statistics | ネットワーク統計。この情報には以下が含まれます。インターフェイス IP アドレス 送信したパケット数 受信したパケット数 送信したバイト数 受信したバイト数 リンク速度 MB/秒 | Agent |
| Number of Prepared Transactions | 準備されたトランザクションの数を保存します。 | Server |
| Number of WAL Files | WAL ファイルの数。 | Server |
| Object Catalog: Database | データベースとそのプロパティのリストを監視します。この情報には以下が含まれます。データベース名 データベースエンコードタイプ データベースがユーザー接続またはシステム接続を許可するかどうか | Server |
| Object Catalog: Foreign Key | 外部キーとそのプロパティのリストを監視します。情報には次のものが含まれます。外部キーを含むテーブルの名前 外部キーが参照するテーブルの名前 テーブルが存在するデータベースの名前 テーブルが存在するスキーマの名前 | Schema |
| Object Catalog: Function | ファンクションとそのプロパティのリストを監視します。情報には次のものが含まれます。ファンクションの名前 ファンクションが存在するスキーマの名前 ファンクションが存在するデータベースの名前 | Schema |

次のページに続く

表 11 – 前のページからの続き

| Name | Information monitored by probe | Level |
|---|---|---|
| Object Catalog: Index | インデックスとそのプロパティのリストを監視します。インデックスの名前 インデックスが関連付けられているテーブルの名前 インデックス付きテーブルが存在するデータベースの名前 | Schema |
| Object Catalog: Schema | スキーマとそれに関連するデータベースとサーバーのリストを監視します。 | Database |
| Object Catalog: Sequence | シーケンスとそのプロパティのリストを監視します。 | Schema |
| Object Catalog: Table | テーブル情報のリストを監視します。情報には以下が含まれます。テーブル名 テーブルが存在するスキーマの名前 スキーマが存在するデータベースの名前 テーブルに主キーがあるかどうかを示すブールインジケーター | Schema |
| Object Catalog: Tablespace | テーブルスペースのリストを監視します。 | Server |
| Operating System Information | オペレーティングシステムの詳細と起動時間。 | Agent |
| Package Catalog | 現在インストール可能なパッケージ。収集される情報には次のものが含まれます。パッケージ名 パッケージバージョン | Agent |
| PG HBA Conf | `pg_hba.conf` ファイルからの認証構成情報。 | Server |
| Server Information | サーバー情報。 | Server |
| Session Information | セッション情報。情報には次のものが含まれます。セッションユーザーの名前 セッションがサーバーに接続した日時 情報が収集された時点のセッションのステータスアイドル、待機中など クライアントアドレスとポート番号 | Server |
| Settings | GUC 変数に現在割り当てられている値を監視します。 | Server |
| SQL Protect | サーバーを監視し、 SQL インジェクション攻撃に関する情報を取得します。 | Server |
| Slony Replication | Slony を使用して複製されたクラスターのラグデータ。 | Database |
| Streaming Replication | ストリーミングレプリケーションを使用しているクラスターを監視し、以下に関する情報を取得します。送信された Xlog ロケーションバイト単位書き込み Xlog ロケーションバイト単位フラッシュ Xlog ロケーションバイト単位再生 Xlog ロケーションバイト単位 Xlog ラグ単位 Xlog ラグページ単位 | Server |

次のページに続く

表 11 – 前のページからの続き

| Name | Information monitored by probe | Level |
|---|---|---|
| Streaming Replication Lag Time | ストリーミングレプリケーションを使用しているクラスターを監視し、以下に関する遅延情報を取得します。レプリケーション遅延時間秒単位 レプリケーションの現在のステータス実行中/一時停止 | Server |
| Streaming Replication Database Conflicts | ストリーミングレプリケーションを使用しているデータベースを監視し、発生した競合に関する情報を取得します。これには、次の数のためにキャンセルされたクエリに関する情報が含まれます ドロップテーブルスペースの競合 ロックタイムアウトの競合 古いスナップショットの競合 固定されたバッファの競合 デッドロックの競合 | Server |
| Table Bloat | 現在のテーブルの肥大化に関する情報。情報には次のものが含まれます。テーブルの名前 テーブルが存在するスキーマの名前 推定ページ数 推定無駄なページ数 行ごとの推定バイト数 | Database |
| Table Frozen XID | 各テーブルのフリーズした XID を監視します。 | Schema |
| Table Size | テーブルサイズに関する情報。テーブルサイズ MB 単位 合計インデックスサイズ MB 単位 合計テーブルサイズ、インデックスと TOAST を含む MB 単位 | Database |
| Table Statistics | テーブル統計。この情報には以下が含まれます。シークエンシャルスキャンの数 シークエンシャルスキャン行の数 インデックススキャンの数 インデックススキャン行の数 挿入行数 更新された行数 削除された行数 ライブ行数 デッド数 rows 最後の VACUUM 最後の auto-vacuum 最後の ANALYZE 最後の自動分析 ANALYZE によって推定されたページ数 ANALYZE によって推定された行数 | Database |
| Tablespace Size | テーブルスペースとそのサイズのリスト。 | Server |
| User Information | 現在のユーザーのリスト。保存される情報には次のものが含まれます。ユーザー名 ユーザータイプスーパーユーザーまたは非スーパーユーザー ユーザーが接続しているサーバー | Server |
| WAL Archive Status | WAL アーカイブのステータス。保存される情報には以下が含まれます。完了した WAL アーカイブの数 保留中の WAL アーカイブの数 最後のアーカイブ時間 失敗した WAL アーカイブの数 最後の障害の時間 | Server |
| xDB Replication | xDB レプリケーションを使用してレプリケートされたクラスターの遅延データ。 | Database |

次のページに続く

表 11 – 前のページからの続き

| Name | Information monitored by probe | Level |
|---|---|---|
| Multixact ID Exhaustion (Wraparound) | multixact ID に関する情報。保存される情報には次のものが含まれます。データベースの名前 データベース内の最も古いマルチ xact ID ラップアラウンドの割合 緊急自動バキュームの割合 | Server |
| Replication Slots | レプリケーションスロットに関する情報。保存される情報には以下が含まれます。レプリケーションスロットの一意のクラスター全体の識別子。この論理スロットが使用している出力プラグインを含む共有オブジェクトのベース名、または物理スロットの場合は null。スロットタイプ 物理的または論理。このスロットが関連付けられているデータベースの OID、または null。論理スロットのみに、関連するデータベースがあります。このスロットが関連付けられているデータベースの名前、または null。論理スロットのみに、関連するデータベースがあります。このスロットが現在アクティブに使用されている場合、Active フィールドは True です。このスロットがデータベースで保持する必要がある最も古いトランザクション。 VACUUM は、後のトランザクションによって削除されたタプルを削除できません このスロットがデータベースで保持する必要があるシステムカタログに影響を与える最も古いトランザクション。 VACUUM は、後のトランザクションによって削除されたカタログタプルを削除できません。最も古い WAL のアドレス LSN。このスロットのコンシューマにまだ必要とされる可能性があるため、チェックポイント中に自動的に削除されません。 | Server |

### Postgres 分散 PGD プローブ

ダッシュボードを使用して PGD グループをモニタするには、次のプローブを有効にする必要があります。これらのプローブを拡張レベルで設定します。

`pgd_superuser` 権限を持つユーザーは、次のすべてのプローブからの情報を表示できます。4.0 より前のバージョンの EDB Postgres Distributed を使用している場合、これらのプローブはすべて EDB Postgres Distributed Enterprise Edition で動作します。

| Probe name | Information monitored by probe |
|---|---|
| PGD Analytic Slot Name | 分析スロット名を提供します。このプローブは、PGD バージョン 6 以降から利用できます。 |

次のページに続く

表 12 – 前のページからの続き

| Probe name | Information monitored by probe |
|---|---|
| PGD Conflict History Summary | 競合タイプごとの行の競合に関する情報。保存される情報には次のものが含まれます。競合の現地時間 競合のタイプ |
| PGD Global Locks | PGD グループのグローバルロックに関する情報。保存される情報には以下が含まれます。グローバルロックが発生したノードの名前 ロックを保持しているプロセスの PID ロックの種類 DDL または DML ロックされたリレーションの名前 DML ロックの場合またはキーアドバイザリロックの場合ロック取得プロセスの内部状態 同じグローバルロックを待機するバックエンドのリスト グローバルロックの取得がオリジンノードによって開始された時間 ローカルノードがローカルロックの取得を試行し始めた時間 acquire_stage が最後に変更された時間 |
| PGD Group CAMO Details | PGD グループの CAMO に関する情報。保存される情報には以下が含まれます。ノードの名前 このノードがパートナーであるノード名 このノードがオリジンであるノード名 接続ステータス 準備状況 保留中または未解決の CAMO トランザクションの数 最後に適用された WAL ログの LSN 最後に受信した WAL ログの LSN |
| PGD Group Replication Slot Details | EDB Postgres 分散グループのレプリケーションスロットに関する情報。保存される情報には次のものが含まれます。EDB EDB Postgres 分散グループの名前オリジンノードの名前ターゲットノードの名前このサブスクリプションが使用するオリジンノードのスロット名アクティブステータスレプリケーションの状態キャッチアップ、ストリーミング、切断,..) 報告された書き込みのおおよそのラグタイム 報告されたフラッシュのおおよそのラグタイム 報告されたリプレイのおおよそのラグタイム send_lsn と現在の WAL 書き込み位置のバイト差 write_lsn と現在の WAL 書き込み位置のバイト差 flash_lsn 間のバイト差と現在の WAL 書き込み位置 |
| PGD Group Subscription Summary | EDB Postgres Distributed グループのサブスクリプションの概要に関する情報。保存される情報には以下が含まれます。サブスクリプションのオリジンの名前 サブスクリプションのターゲットの名前 最後に再生されたトランザクションのタイムスタンプ 現在と最後に再生されたトランザクションの時間のラグ |
| PGD Monitor Group Raft | クラスター全体の raft チェックのステータスとメッセージ。 |
| PGD Group Raft Details | EDB Postgres 分散グループ内のすべてのノードからの raft コンセンサスステータスに関する情報。保存される情報には以下が含まれます。ノードの名前 ノードの raft ワーカー状態 RAFT_LEADER のノード ID raft 選挙の内部 id raft スナップショットの内部 id |
| PGD Monitor Group Versions | クラスタ全体のバージョンチェックのステータスとメッセージ。 |
| PGD Group Versions Details | EDB Postgres Distributed グループ内の各ノードにインストールされている Postgres、pglogical、 EDB Postgres Distributed のバージョンの詳細に関する情報。保存される情報には以下が含まれます。ノードの名前 ノードにインストールされている Postgres バージョン ノードにインストールされている pg 論理バージョン ノードに分散されている EDB Postgres のバージョン 4.0 より前のバージョンの EDB Postgres 分散エディション Standard/Enterprise 。 |

次のページに続く

表 12 – 前のページからの続き

| Probe name | Information monitored by probe |
|---|---|
| PGD Node Group Config Summary | 以下を含むグループ構成に関する情報 グループ ID グループオプション名 グループの現在値 グループの現在値のソース このプローブは、バージョン 6 以降で利用できます。 |
| PGD Node Replication Rates | 特定のノードからの発信レプリケーションアクティビティに関する情報。保存される情報には以下が含まれます。ターゲットピアノードの名前 最新の送信位置 再生されたとして報告された最新の位置 報告された再生のおおよそのラグタイム attempts * lsn とオリジン上の現在の WAL 書き込み位置のバイト差 両者間の人間が判読可能なバイト差 resume_lsn と現在の WAL 書き込み位置 ピアノードがまだ適用されていないすべての変更に追いつくまでに必要なおおよその時間 **注:** このプローブには、EDB Postgres Advanced Server または EDB Postgres ExtendedPGE が必要です。コミュニティ PostgreSQL ではサポートされていません。コミュニティ PostgreSQL の PGD クラスターに対して実行すると、*apply_rate* **`および`** *`catchup_interval`* **`メトリックは使用できず、エージェントログは`** *`lag tracking is not supported with this variant of Postgres`* **`をレポートします。基本的なレプリケーションラグメトリック** *`replay_lag`*、`replay_lag_bytes`は、すべての PostgreSQL バリアントで_PGD Node Slots*プローブを介して利用できます。 |
| PGD Node Roles | PGD クラスター内のノードのロールに関する情報。考えられるロール値の一部は次のとおりです。 RAFT_LEADER RAFT_FOLLOWER RAFT_CANDIDATE 書き込みリーダー 書き込み候補 |
| PGD Node Slots | ローカル PGD データベースノードのレプリケーションスロット、そのステータス、およびレプリケーションの進行状況へのマッピングに関する情報。保存される情報には以下が含まれます。スロットの名前 ターゲットノードの名前 PGD グループの名前 ターゲットノード上のデータベース名 スロットに接続されているプロセスの PID スロットに必要なカタログ XID の IP アドレス クライアント接続 最後に送信された位置 再生されたとして報告された最新の位置 報告された再生のおおよそのラグタイム attempts_lsn と現在の WAL 書き込み位置のバイト差 attempts_lsn と現在の WAL 書き込み位置の人間が判読可能なバイトの差 |
| PGD Node Summary | PGD グループ内のすべてのノードに関する情報。保存される情報には以下が含まれます。ノードの名前 ノードが属している PGD グループの名前 人間が読める形式のノードの一貫した状態 ノードが到達しようとしている状態参加または昇格中に の名前サブスクライブされた repsets ユニバーサル一意 ID、バージョン 6 以降 |
| PGD Workers | EDB Postgres 分散ノードのワーカーに関する情報。保存される情報には以下が含まれます。ワーカープロセスの PID ワーカークエリ開始タイムスタンプ ワーカー状態変更タイムスタンプワーカー待機イベントタイプ ワーカー待機イベントワーカー状態ワーカーロール名ワーカーコミットタイムスタンプワーカーローカルタイムスタンプ オリジンの名前ノード 受信 LSN 受信コミット LSN 最後の正確なリプレイ LSN 最後の正確なフラッシュ LSN 最後の正確なリプレイタイムスタンプワーカークエリー |

次のページに続く

表 12 – 前のページからの続き

| Probe name | Information monitored by probe |
|---|---|
| PGD Work Errors | EDB Postgres 分散データベースノードの作業エラーに関する情報。保存される情報には以下が含まれます。エラーを発生させたワーカーのプロセス ID ノードが一部である EDB Postgres 分散グループの名前 オリジンノードの名前 ソースノードの名前 ターゲットノードの名前 サブスクリプションの名前 このワーカーのロールの内部識別子 このワーカーのロールの名前 エラーの日時 エラーの経過時間 エラーの説明 エラーが発生したコンテキスト リモートリレーション ID サブスクリプションライタ ID サブスクリプライタの名前 |

### フェールオーバーマネージャープローブ

フェールオーバーマネージャーの情報とステータスを監視するには、次のプローブを有効にする必要があります。

| Probe name | Information monitored by probe | Level |
|---|---|---|
| Failover Manager Cluster Info | フェールオーバーマネージャークラスターを監視し、クラスターに関する情報を結果ます。このプローブは、フェールオーバーマネージャーバイナリのクラスター名前とパスがサーバーのプロパティダイアログボックスで提供されている場合に有効になります。 | Server |
| Failover Manager Node Status | フェールオーバーマネージャークラスターを監視し、クラスター内の各ノードに関する詳細を結果ます。このプローブは、フェールオーバーマネージャーバイナリのクラスター名前とパスがサーバーのプロパティダイアログボックスで提供されている場合に有効になります。 | Server |

### Patroni プローブ

Patroni クラスター情報とステータスをモニタするには、次のプローブを有効にする必要があります。

| Probe name | Information monitored by probe | Level |
|---|---|---|
| Patroni Cluster Status | Patroni クラスターを監視し、以下に関する情報を取得します。 Patroni クラスター名 クラスターが動作している現在の PostgreSQL タイムライン ID プライマリノード名 現在のプライマリノードのホスト名または IP アドレス DCS の正常なステータスを追跡するためのブールインジケーター Patroni クラスターのダウンステータスを追跡する クラスターの一時停止ステータスを追跡するためのブールインジケーター | Server |

次のページに続く

表 14 – 前のページからの続き

| Probe name | Information monitored by probe | Level |
|---|---|---|
| Patroni Node Status | Patroni クラスターのノードを監視し、以下に関する情報を取得します。 Patroni クラスター名 ノード名 ノードのホスト名または IP アドレス ノードに現在割り当てられているロール。使用可能な値は`leader`、*replica*、または`standby_leader`です。ノードのステータス。使用可能な値は`running`、*streaming*、*starting*、または`stopped`です。ノードが現在いるタイムライン MB 単位のレプリケーションラグタイム | Server |

### Managing and customizing probes

プローブは、特定の監視対象オブジェクトに関する一連のパフォーマンスメトリックを返すスケジュールされたタスクです。プローブは、監視対象サーバー、データベース、データベースオブジェクト、オペレーティングシステム、またはエージェントから統計を取得します。

**Manage Probes** タブでは、管理者は Postgres Enterprise ManagerPEM エージェントがデータベースとサーバーを監視するために使用するデータ収集メトリックプローブを構成できます。このインターフェイスは、どのデータが収集されるか、頻度、および保持される期間を詳細に制御します。

#### プローブ概要表

[ **Manage Probes** ] タブには、次の構成可能な属性を持つ使用可能なプローブのリストが表示されます。

- **Name** - プローブの説明的な識別子たとえば、データベース統計、テーブルスペースサイズ。
- **収集方法** - `SQL`、`WMI`、または `Batch/Shell Script` などのデータの収集に使用されるプロトコル。
- **ターゲットタイプ** - `Server`、`Database`、`Table`、`Schema`、`Index`、`Sequence`、`Function/Procedure` などの監視対象階層のレベル。
- **モニタリングターゲット** - `Database Server`、`Database`、`Schema`、または `Table` など、プローブが適用される親エンティティ。
- **間隔分/秒** - プローブが実行される頻度たとえば、5 分または 30 分ごと。
- **Enabled?** - プローブが有効か無効かを表示するトグルスイッチ。
- **データ保持** - 収集された履歴データがパージされるまで PEM データベースに保存される日数。

#### キー管理アクション

アイコンまたはバーを使用して、次のタスクを実行できます。

- **フィルター/検索** - 上部の検索バーを使用して、特定のプローブを名前で見つけます。
- **Toggle System Probes** - 右上の **Show System Probes?** スイッチを使用して、ビルトインのデフォルトプローブを表示または非表示にします。

- **カスタムプローブの作成** - 右上のツールバーのプラスアイコンを使用して、新しいカスタムプローブを作成します。
- **プローブリストの更新** - 右上のツールバーの更新アイコンを使用して、プローブのリストをリロードし、最近の変更を反映します。
- **Save Changes** - 右上のツールバーのディスクアイコンを使用して、プローブ構成に加えた変更を保存します。
- **エクスポート/インポート** - 右上のツールバーの送信ボックス矢印および受信ボックス矢印アイコンを使用して、プローブ構成をエクスポートするか、ファイルからプローブ構成をインポートします。
- **プローブの編集** - プローブ名の左側にある鉛筆アイコンを使用して、プローブの収集間隔または保持期間を変更します。
- **プローブの削除** - ゴミ箱アイコンを使用して、カスタムプローブを削除します。ビルトインシステムプローブは削除できません。

> **注釈**
>
> 無効になったプローブからのデータがチャートで使用されている場合、チャートの左上隅に情報アイコンが表示され、提供されたリンクをクリックしてプローブを有効にできます。

### Creating custom probes

**Manage Probes** タブを使用して、新しいカスタムプローブを作成したり、既存のカスタムプローブを変更したりできます。

**Manage Probes** タブから、右上のツールバーのプラスアイコンを選択してカスタムプローブを作成します。

1. **General** タブを使用して、新しいプローブのプロパティを指定します。

- **Name** フィールドを使用して、新しいプローブの名前を提供します。
- **Collection Method** フィールドを使用して、プローブタイプを指定します。リストから、
- SQL - プローブは、 SQL ステートメントを介して情報を収集します。
- WMI - プローブは、 Windows Management Instrumentation 拡張機能を介して情報を収集します。
- Batch/Shell Script - プローブはコマンドスクリプトを使用します情報を収集するためのシェルスクリプト。

Linux システムでバッチプローブを作成する前に、`agent.cfg` ファイルを変更し、 `allow_batch_probes` パラメーターを `true` に設定する必要があります。次に、PEM エージェントを再起動します。`agent.cfg` ファイルは`/usr/edb/pem/agent/etc/` ディレクトリにあります。

Windows システムでは、エージェント設定はレジストリに保存されます。バッチプローブを作成する前に、`AllowBatchProbes` レジストリエントリのレジストリエントリを変更し、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\EnterpriseDB\PEM\agent` にあります。

バッチプローブはプラットフォーム固有です。 **Batch** の収集方法を指定する場合は、 **Platform** フィールドでプラットフォームタイプを指定する必要があります。

- Linux システムでスクリプトを呼び出すには、`agent.cfg` の `batch_script_user` パラメーターのエントリーを変更する必要がありますファイルを作成し、スクリプトの実行に使用するユーザーを指定します。このパラメーターには、非 root ユーザーまたは root を指定できます。ユーザーを指定しない場合、または指定され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。
- **Target type** リストを使用して、モニタするプローブのオブジェクトタイプを選択します。**収集方法**が WMI の場合、**ターゲットタイプ**は無効になります。プローブで **ターゲットタイプ**を構成して、収集されたメトリックがシステム階層の目的のレベルに関連し正確であることを確認します。ターゲットタイプを設定することにより、テーブルやインデックスなどの個々のデータベースオブジェクトからサーバー全体の監視に至るまで、さまざまなコンポーネントを効果的に監視できます。

ターゲットタイプ構成マトリックス

次の表は、ターゲットタイプの構成、その監視スコープ、およびそれらの構成のための必須フィールドの概要を提供します。

このテーブルでは

- **Target type** 列は、プローブでモニタするオブジェクトを決定します。
- **Executionlevel** 列は、モニタリングプローブをデータベースごと、サーバーごと、エージェントごとに 1 回実行するかどうかを決定します。拡張機能レベルごと。
- **必須列**列は、必要なデータが収集されるようにプローブクエリーで構成する必要がある列を示します。
- **Probe attempts** 列は、理解を深めるために探索できる既存のプローブをいくつか提供します。プローブが実際にどのように使用されるか。

| Target type | Execution level | Mandatory columns | Probe examples |
|---|---|---|---|
| Agent | Agent | None | cpu_usage |
| Server | Server | None | server_info |
| Database | Database | None | database_size |
| Schema | Database | schema_name | oc_extension |
| Table | Database | schema_name, table_name | table_size |
| Index | Database | schema_name, index_name | index_size |
| Sequence | Database | schema_name, sequence_name | oc_sequence |
| View | Database | schema_name, view_name | mview_size |

次のページに続く

表 15 – 前のページからの続き

| Target type | Execution level | Mandatory columns | Probe examples |
|---|---|---|---|
| Function | Database | schema_name, arg_types, function_type, function_name | function_statistics |
| Extension | Extension | None | bdr_node_summary |

**!!!note`2_blank_2_-` データベース以上のターゲットタイプスキーマ、テーブル、インデックス、ビュー、シーケンス、ファンクションを含むカスタムプローブは、データベースレベルで情報を収集します。**
***1_blank_2_-* スキーマ以上のターゲットタイプに設定されたシステムプローブスキーマレベルまでの情報を収集します。 .. ::**
</details>

- **Minutes** および **Seconds** セレクターを使用して、プローブがデータを収集する頻度を指定します。
- **Enabled?** トグルボタンを使用して、プローブをデフォルトで有効にするか無効にするかを指定します。プローブを有効にすると、指定された間隔でデータの収集が開始されます。プローブを無効にすると、有効にするまでデータは収集されません。

**注釈**

無効になったプローブからのデータがチャートで使用されている場合、チャートの左上隅に情報アイコンが表示され、提供されたリンクをクリックしてプローブを有効にできます。

- **Data Retention** フィールドを使用して、収集された情報がプローブの履歴テーブルに保持される日数を指定します。
- **Discard fromhistory** トグルボタンを使用して、サーバーがプローブの履歴テーブルを作成するかどうかを指定します。プローブ履歴を破棄する場合は **Yes** を選択し、テーブルにプローブ履歴を保持する場合は **No** を選択します。
- **Platform** リストを使用して、プローブがモニタするプラットフォームのタイプを指定します。このフィールドは、**Collection** 方法が **Batch** の場合にのみ有効になります。

2. **Columns** タブを使用して、プローブデータを保存する列を定義します。新しい列を定義するには
3. **Columns** タブに移動し、右上隅のプラスアイコンを選択して列を追加します。
4. **Name** フィールドに、列の名前を入力します。
5. **を選択します新しい列名の左側を編集して、列に関する情報を提供します。

```
-** Name **フィールドに列の説明的な名前を入力します。

-** Internal name **フィールドは有効になっていませんカスタムプローブの場合.

-** Column type **リストを使用して、列がキー列主キー列であるか、非キー列であるかを指定します。非キー列は、一般にメトリック項目グラフ化できる値です。
```

(次のページに続く)

(前のページからの続き)

-** Data type **リストを使用して、列に保存するデータのタイプを指定します。

-** Unit **フィールドを使用して、列に格納されているメトリックに適用される測定単位。この単位は、カスタムチャートまたは Capacity Manager チャートの y 軸に表示されます。これはオプショナルフィールドです。

-** Graphable? **トグルボタンを使用して、定義したメトリックをグラフ化できるかどうかを指定し、Capacity Manager または Manage Charts ダイアログボックスからプローブにアクセスできるようにします。

-** Is PIT?を使用します**トグルボタンでメトリックをポイントインタイムで保存するかどうかを指定します。

ポイントインタイムメトリックは、任意の時点で変化します増加または減少します。たとえば、データベースサイズはポイントインタイムメトリックです。任意の時点で、データベースのサイズは変動します。ポイントインタイムではないメトリック累積メトリックとも呼ばれ、そのサイズは時間の経過とともに常に増加します。たとえば、Blocks Read と Tuples Read は累積メトリックです。値は同じまたは増加します。

- **Calculate PIT?** トグルボタンを使用して、サーバーがメトリックデータのポイントインタイム値を計算するように指定します。 **Is PIT** が **Yes** の場合、 **PIT の計算** は無効になっています。

PEM では、累積メトリックのポイントインタイム値を保存することもできます。PEM は、現在の値から累積メトリックの最後に収集された値を減算し、差をポイントインタイム値として保存します。

3. **Code** タブを使用して、実行するプローブのデフォルトコードを指定します。

- プローブが SQL プローブの場合、**コード**タブでプローブによって呼び出される **SQL SELECT** ステートメントを指定する必要があります。クエリによって返される列名は、[ **列** ] タブで指定された **内部名**と一致する必要があります。クエリによって返される列の数、列名、データ型などは、**Columns** タブで指定された情報と一致する必要があります。
- プローブがバッチプローブの場合、プローブの実行時に呼び出すシェルまたは.bat スクリプトを指定する必要があります。スクリプトの出力は次のとおりです。
- 最初の行には、 **Columns** タブに提供される列の名前が含まれている必要があります。各列名はタブ t 文字で区切る必要があります。2 行目以降、各行には、各列のデータをタブ文字で区切って含める必要があります。
- 指定された列がキー列として定義されている場合、スクリプトが行をまたがってその列の重複データを生成しないことを確認します出力の。[ **Columns** ] タブで指定された列の数とその名前、データ型などは、スクリプト出力の出力と一致する必要があります。
- プローブが WMI プローブの場合、WMI クエリーを **SELECT WMI** クエリーとして指定する必要があります。 **SELECT** ステートメントで参照される列名は、 **Column** タブで指定された対応する列の名前と同じである必要があります。クエリによって返される列名は、[ **列** ] タブで指定された **内部名**と一致する必要があります。クエリによって返される列の数、列名、データ型などは、**Columns** タブで指定された情報と一致する必要があります。

4. **Alternate Code** タブを使用して、サーバーの特定のバージョンでプローブが起動した場合に呼び出すコードを提供します。バージョン固有のコードを提供するには、**Applies to all Database server version?** スイッチを **No** に移動し、右上隅のプラスアイコンを選択して特定のバージョンを追加します。次に、**Server Version(s)** リストを使用してバージョンを選択し、 **Edit** バージョン名の左側を選択して、そのバージョンでプローブが実行されたときに実行するコードを提供します。

データベースバージョンを選択し、 **プローブコード**列を空白のままにすると、PEM は、そのバージョンと一致するサーバーでプローブが実行されるときに、**コード** タブで指定されたコードを呼び出します。

プローブの定義が終了したら、[ **カスタムプローブ** ] タブの隅にある**保存**を選択して定義を保存し、プローブデータをカスタムチャートやグラフで使用できるようにします。

> **注釈**
>
> 重複エントリによるエージェントの障害を防ぐには、カスタムプローブクエリーが一意の行を返すようにしてください。クエリが同じターゲットに対して複数の行を返す場合、少なくとも 1 つのキー列を定義して、各行の一意の複合識別子を作成する必要があります。

### Exporting or importing probes

**Manage Probes** タブからプローブ定義をエクスポートおよびインポートして、さまざまな PEM 環境間でプローブを共有できます。プローブをエクスポートすると、PEM はプローブ定義を含む JSON ファイルを生成します。このファイルを使用して、プローブを別の PEM 環境にインポートできます。生成された JSON ファイルをテンプレートとして使用して、新しいプローブを作成することもできます。

プローブをエクスポートするには

1. Probes テーブルからプローブを選択します。
2. テーブルの右上隅にある送信ボックスの矢印アイコンを選択します。
3. **Save File** を選択し、 **OK** を選択します。プローブ定義を含む JSON ファイルがローカルマシンにダウンロードされます。

プローブをインポートするには

1. テーブルの右上隅にある受信ボックスの矢印アイコンを選択します。
2. **Browse** を選択して、インポートするプローブコードを含む JSON ファイルを選択します。
3. **Import** を選択します。

インポートするファイルを選択した後、**Skip stable** チェックボックスを選択できます。このオプションは、プローブが既に存在する場合はスキップし、インポートがスキップされたことを通知するメッセージを表示します。

- チェックボックスを選択せず、プローブが既に存在する場合、プローブはインポートされず、エラーメッセージが報告されます。

> **注釈**

**Import**は、構成された保持ポリシーに従って履歴データを保持するように構成されている場合があるため、既存のプローブを上書きできません。

- チェックボックスを選択せず、プローブは存在しないが、pem スキーマの対応するテーブルが存在する場合、同じテーブルを使用してプローブは正常にインポートされます。

**注釈**

プローブが削除され、**Manage Probes**タブにリストされず、そのプローブのデータを保持するテーブルが `pem` スキーマに存在する可能性があります。

### Deleting a custom probe

削除できるのはカスタムプローブのみです。プローブを削除するには

1. **Manage Probes** タブに移動します。
2. プローブテーブルでプローブ名を選択し、プローブ名の左側にあるゴミ箱アイコンを選択します。

プローブを削除すると、PEM はプローブ定義と、そのプローブによって収集されたすべての履歴データを削除します。プローブ履歴は、プローブ定義の**履歴保持**フィールドで指定された時間持続します。削除中に、プローブ定義が削除され、対応するテーブルが pemdata および pemhistory スキーマから削除されます。

システムプローブは、PEM が提供するビルトインプローブであり、PEM スキーマの一部です。システムプローブのみを変更できます。システムプローブを削除しようとすると、PEM はエラーを報告します。

### Managing probes using pemworker

Postgres Enterprise ManagerPEM バージョン 9.6 から、--probe-display-name 、--probe-internal-name 、または--probe-id オプションを使用して pemworker ユーティリティを使用して、プローブを有効または無効にできます。

#### 例

**Linux の場合**

次の例は、プローブ表示名を使用してプローブを無効にする方法を示しています。

```
export PEM_SERVER_PASSWORD=edb
./pemworker --pem-user enterprisedb --disable-probe --probe-display-name "cpu usage"
__OUTPUT__
Wed Mar 27 01:43:40 2024 INFO: Probe status updated for agent (id: 1)
```

この例は、プローブ表示名を使用してプローブを有効にする方法を示しています。

```
export PEM_SERVER_PASSWORD=edb
./pemworker  --pem-user enterprisedb --enable-probe --probe-display-name "database␣
↪size"
```

(次のページに続く)

(前のページからの続き)

```
__OUTPUT__
Wed Mar 27 02:02:20 2024 INFO: Probe status updated for server (id: 1)
```

**Windows の場合*

次の例は、プローブ表示名を使用してプローブを無効にする方法を示しています。

```
set PEM_SERVER_PASSWORD=edb
./pemworker.exe DISABLE-PROBE --pem-user postgres --probe-display-name "CPU usage"
__OUTPUT__
INFO: Probe status updated for agent (id: 1)
```

この例は、プローブ表示名を使用してプローブを有効にする方法を示しています。

```
set PEM_SERVER_PASSWORD=edb
./pemworker.exe ENABLE-PROBE --pem-user postgres --probe-display-name "CPU usage"
__OUTPUT__
INFO: Probe status updated for agent (id: 1)
```

### Probe configuration

**Probe Configuration** 設定は、Postgres Enterprise ManagerPEM インターフェイス内でプローブを変更するときに使用できます。

**Management > Probes > Probe configuration** に移動して、これらの設定にアクセスできます。

- **Name** - 特定のプローブの説明的な名前たとえば、データベース統計、テーブルスペースサイズ。
- **Probe enabled?** - プローブの状態を切り替えるドロップダウンメニュー。 `Default` を選択すると、PEM エージェントのグローバル構成セットに従います。`Enable` を選択すると、グローバル設定に関係なくプローブが有効になりますが、`Disable` は無効にします。
- **ステータス** - プローブが現在有効か無効かを示します。
- **緑色のチェックマーク** - プローブはアクティブで有効です。
- **赤い十字マーク** - プローブは非アクティブで無効になっています。
- **実行頻度** - 分と秒で定義されたプローブが実行される間隔。
- **Usedefault?** - PEM エージェントレベルで定義されたデフォルトの実行頻度をプローブが使用するかどうかを決定するトグル。無効の場合、カスタム間隔を指定できます。
- **Time (HH:MM:SS)** - デフォルトが無効の場合、プローブ実行の特定の時間間隔を手動で設定できます。
- **履歴保持** - プローブの履歴データが PEM データベースに保持される日数。
- **Usedefault?** - プローブがで定義されたデフォルトのデータ保持期間を使用するかどうかを決定するトグル。PEM エージェントレベル。無効の場合、カスタム保持期間を指定できます。
- **日** - デフォルトが無効の場合、このプローブの履歴データを保持する日数を手動で設定できます。

#### アクション

- **Reset** - プローブの構成設定をグローバルレベルで定義されたデフォルト値にリセットするボタン。これにより、このプローブに適用したカスタム設定がオーバーライドされます。
- **Copy probes** - 選択したオブジェクトから他のオブジェクトに現在のプローブの構成をコピーできるボタン。これは、各プローブを個別に構成することなく、複数のプローブに同じ設定をすばやく適用する場合に役立ちます。

> **注釈**
>
> プローブ構成に加えられた変更はすぐには反映されません。これらの変更が完全に反映され、システム全体で同期されるまでに数秒かかる場合があります。

### Copying probe configurations

**Copy Probe Configuration** ダイアログボックスを使用して、1 つの監視対象オブジェクトから同じタイプの 1 つ以上の監視対象オブジェクトにプローブ定義をコピーできます。**Copy Probe Configuration** ダイアログボックスを開くには

1. PEM クライアントツリーで、プローブをコピーするオブジェクトを選択します。
2. **Management > Probes** を選択します。
3. **Copy Probe Configurations** を選択します。

**Copy Probe Configurations** ダイアログボックスが開き、 **Copy Probe Configuration** ダイアログボックスを開いたオブジェクトからツリーコントロールで選択した場所にプローブ定義をコピーします。

**Copy Probe Configuration** ツリーで親ノードを指定すると、PEM は、ツリー内のそのノードの下にある同じタイプの各オブジェクトにプローブ構成をコピーします。たとえば、1 つのスキーマからデータベース内にあるすべてのスキーマにプローブ定義をコピーするには、ターゲットスキーマの親データベースのみを選択します。そのオブジェクトがコピーされるプローブのソースである場合、リストされたターゲットオブジェクトの名前の左側に赤い警告記号が表示されます。

ターゲットオブジェクトを選択したら、[ **Configure Probes** ] を選択して、ダイアログボックスで選択した場所にプローブ定義をコピーします。

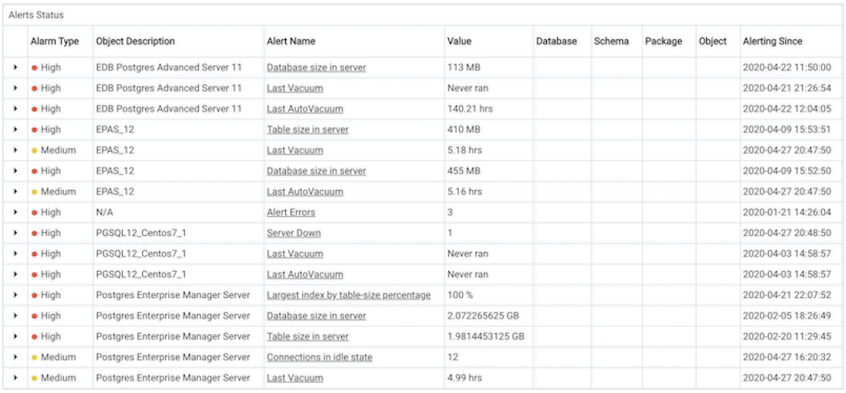
> **注釈**
>
> コピープローブは、拡張レベルのプローブではサポートされていません。

## 8.3.2 Alerts

PEM は、登録されたサーバーを継続的に監視します。パフォーマンスメトリックを、各統計について良好または許容可能なパフォーマンスを指定する事前定義されたユーザー指定のしきい値と比較します。許容可能なしきい値からの逸脱は、アラートをトリガーします。アラートは、PEM がシステム統計と比較するシステム定義またはユーザー定義の条件セットです。アラートは、注意が必要な登録サーバーの状態を通知します。

#### 8.3.2.1 グローバルダッシュボードを介したアラートの表示

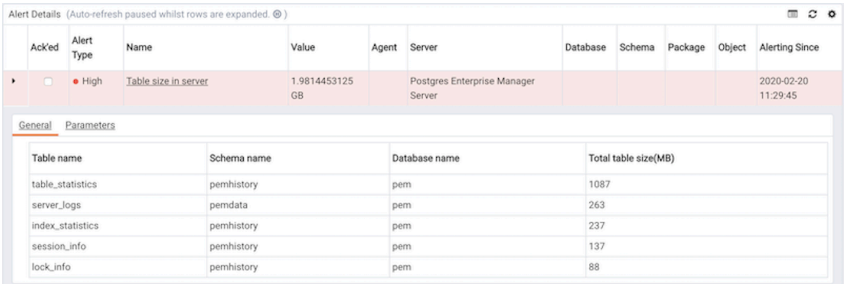
システム統計がその統計に指定された境界から逸脱すると、アラートがトリガーされます。アラートは、グローバル概要ダッシュボードのアラートステータステーブルの左端の列に、重大度の高赤、低黄、または中オレンジの警告を表示します。


Alerts Status

| | Alarm Type | Object Description | Alert Name | Value | Database | Schema | Package | Object | Alerting Since |
|---|---|---|---|---|---|---|---|---|---|
| ▸ | High | EDB Postgres Advanced Server 11 | Database size in server | 113 MB | | | | | 2020-04-22 11:50:00 |
| ▸ | High | EDB Postgres Advanced Server 11 | Last Vacuum | Never ran | | | | | 2020-04-21 21:26:54 |
| ▸ | High | EDB Postgres Advanced Server 11 | Last AutoVacuum | 140.21 hrs | | | | | 2020-04-22 12:04:05 |
| ▸ | High | EPAS_12 | Table size in server | 410 MB | | | | | 2020-04-09 15:53:51 |
| ▸ | Medium | EPAS_12 | Last Vacuum | 5.18 hrs | | | | | 2020-04-27 20:47:50 |
| ▸ | High | EPAS_12 | Database size in server | 455 MB | | | | | 2020-04-09 15:52:50 |
| ▸ | Medium | EPAS_12 | Last AutoVacuum | 5.16 hrs | | | | | 2020-04-27 20:47:50 |
| ▸ | High | N/A | Alert Errors | 3 | | | | | 2020-01-21 14:26:04 |
| ▸ | High | PGSQL12_Centos7_1 | Server Down | 1 | | | | | 2020-04-27 20:48:50 |
| ▸ | High | PGSQL12_Centos7_1 | Last Vacuum | Never ran | | | | | 2020-04-03 14:58:57 |
| ▸ | High | PGSQL12_Centos7_1 | Last AutoVacuum | Never ran | | | | | 2020-04-03 14:58:57 |
| ▸ | High | Postgres Enterprise Manager Server | Largest index by table-size percentage | 100 % | | | | | 2020-04-21 22:07:52 |
| ▸ | High | Postgres Enterprise Manager Server | Database size in server | 2.072265625 GB | | | | | 2020-02-05 18:26:49 |
| ▸ | High | Postgres Enterprise Manager Server | Table size in server | 1.9814453125 GB | | | | | 2020-02-20 11:29:45 |
| ▸ | Medium | Postgres Enterprise Manager Server | Connections in idle state | 12 | | | | | 2020-04-27 16:20:32 |
| ▸ | Medium | Postgres Enterprise Manager Server | Last Vacuum | 4.99 hrs | | | | | 2020-04-27 20:47:50 |

図 3: The Alert Status table

PEM サーバーには、サーバーをアクティブに監視する多数の事前定義されたアラートが含まれています。アラート定義により、アラートの原因に関する詳細を利用できる場合があります。重大度警告の右側にある下矢印を選択して、アラートをトリガーした条件の詳細を含むダイアログボックスを開きます。


Alert Details (Auto-refresh paused whilst rows are expanded.)

| | Ack'ed | Alert Type | Name | Value | Agent | Server | Database | Schema | Package | Object | Alerting Since |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ▸ | ☐ | High | Table size in server | 1.9814453125 GB | | Postgres Enterprise Manager Server | | | | | 2020-02-20 11:29:45 |

General | Parameters

| Table name | Schema name | Database name | Total table size(MB) |
|---|---|---|---|
| table_statistics | pemhistory | pem | 1087 |
| server_logs | pemdata | pem | 263 |
| index_statistics | pemhistory | pem | 237 |
| session_info | pemhistory | pem | 137 |
| lock_info | pemhistory | pem | 88 |

図 4: Alert details

PEM は、カスタマイズされたアラートを作成できるインターフェイスも提供します。各アラートは、アラートテンプレートで定義されたメトリックを使用します。アラートテンプレートは、サーバーがリソースまたはメトリックの統計を評価する方法を定義します。PEM サーバーには、事前定義されたアラートテンプレートが含まれており、カスタムアラートテンプレートを作成できます。

#### 8.3.2.2 アラートダッシュボードを介したアラートの表示

**Dashboards** メニュー **Monitoring** タブを使用して、Alerts ダッシュボードを開きます。アラートダッシュボードには、アクティブなアラートの概要と各アラートのステータスが表示されます。

Alerts ダッシュボードヘッダーには、ダッシュボードが最後に更新された日時と、現在のアラートの数が表示されます。

[Alerts Overview] セクションには、アクティブなアラートの視覚的表現と、現在の高、低、および中アラートの数が表示されます。グラフの左側の垂直バーは、各列に表示されるアラートの数を提供します。バーにカーソルを合わせると、グラフの右上隅に選択したアラートの重大度のアラート数が表示されます。

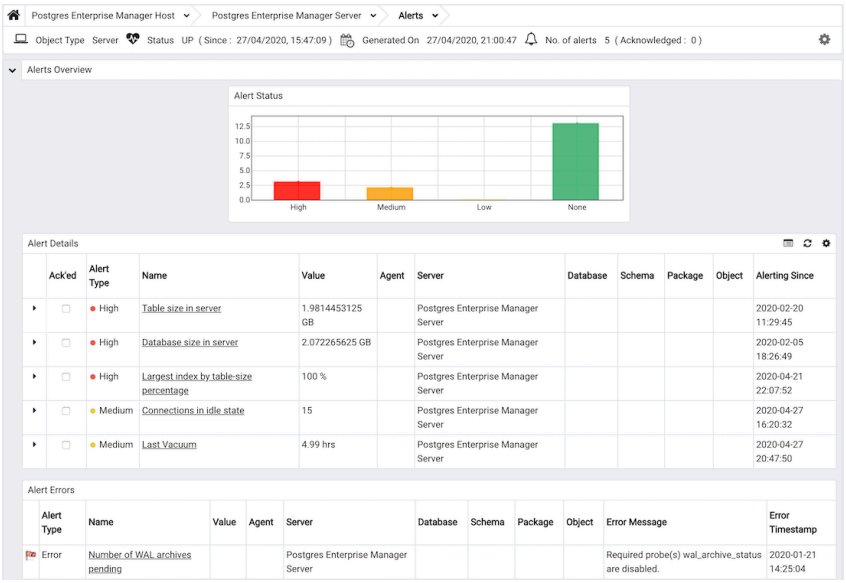



図 5: The Alerts Dashboard

Alert Details テーブルは、現在トリガーされているアラートのリストを提供します。エントリは、重大度の高いものから低いものの順に表示されます。各エントリには、アラートを識別し、アラートをトリガーした条件を認識できる情報が含まれています。アラートを選択して、アラート定義に関する詳細情報を確認します。

Alert Errors テーブルには、必要なプローブを誤って無効にしたり、アラートパラメーターの不適切な構成などの構成関連のエラーが表示されます。[Error Message] 列で提供される情報を使用して、エラーの原因となっている競合を特定して解決できます。

#### 8.3.2.3 アラートダッシュボードのカスタマイズ

アラートダッシュボードに表示されるテーブルとグラフをカスタマイズできます。テーブルまたはグラフをカスタマイズするには、右上隅にある**設定** を選択します。

[ Personalize Chart Configuration ] ダイアログボックスのフィールドを使用して、表示設定を提供します。

- **Auto Refresh** フィールドを使用して、テーブルまたはチャートに表示されるデータの更新の間の秒数を指定します。
- **Download as** フィールドを使用して、チャートを JPEG 画像としてダウンロードするか、PNG 画像としてダウンロードするかを示します。
- **カラーセレクター** を使用して、チャートで使用する色を指定します。

カスタマイズを保存するには、右上隅の **Save** チェックマークを選択します。以前の変更を削除し、デフォルト値に戻すには、**Delete** を選択します。

### Managing alerts

PEM クライアントの **Manage Alerts** タブを使用して、アラートを定義、コピー、または管理します。 [ **Manage Alerts** ] タブを開くには、 **Management > Alerts > Manage Alerts** を選択します。

クイックリンクツールバーを使用して、アラートを管理するためのダイアログボックスとタブを開きます。

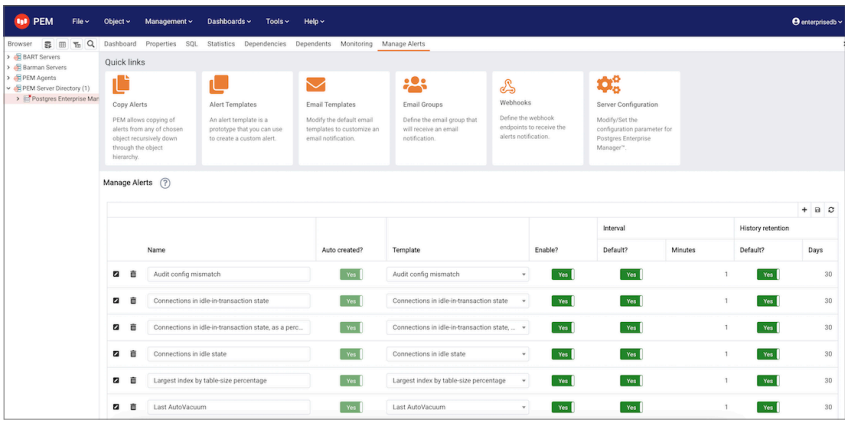



図 6: The Manage Alerts tab

- **Copy Alerts** を選択して Copy Alert Configuration ダイアログボックスを開き、アラート定義をコピーします。
- **Alert Templates** を選択して [ **Alert Template** ] タブを開き、アラートテンプレートを変更または作成します。
- **Send the notifications** を選択して [ Email Template ] ダイアログボックスを開き、デフォルトのメールテンプレートを変更してメール通知をカスタマイズします。
- **Send the notifications** を選択して [ **Send the notifications** ] タブを開き、メールグループを変更または作成します。
- **Webhooks** を選択して [ **Webhooks** ] タブを開き、Webhook エンドポイントを作成または管理します。
- **Server Configurations** を選択して Server Configuration ダイアログボックスを開き、サーバー構成設定を確認または変更します。

**Manage Alerts** タブの Alerts セクションのテーブルを使用して、新しいアラートを作成するか、既存のアラートを管理します。

### Alert templates

アラートテンプレートは、アラートのプロパティを定義するプロトタイプです。アラートは、監視対象オブジェクトの現在の状態をアラートテンプレートで指定されたしきい値と比較して、管理上の注意が必要な状況かどうかを判断するようにサーバーに指示します。

**Alert Templates** タブを使用して、カスタムアラートテンプレートを定義したり、既存のアラートテンプレートの定義を表示したりできます。[ **Alert Templates** ] タブを開くには、**Management > Alerts > Manage Alerts** を選択します。 [ **Manage Alerts** ] タブの [Quick Links] ツールバーで、[ **Alert Templates** ] を選択します。

**Show System Template** リストを使用して、Alert Templates テーブルに表示されるアラートテンプレートをフィルタリングします。リストから PEM 階層のレベルを選択して、そのレベルのすべてのテンプレートを表示します。

### 新しいアラートテンプレートの定義

新しいアラートテンプレートを定義するには、 **Show System Template** リストから **None** を選択します。次に、アラートテンプレートテーブルの右上隅にあるプラス記号+をクリックします。アラートテンプレートエディターが開きます。

**General** タブのフィールドを使用して、テンプレートに関する一般的な情報を指定します。

- **Template name** フィールドを使用して、新しいアラートテンプレートの名前を指定します。
- **Description** フィールドを使用して、アラートテンプレートの説明を提供します。
- **Target type** リストを使用して、アラートの焦点であるオブジェクトのタイプを選択します。
- **Applies to server** リストを使用して、アラートを適用するサーバータイプ EDB Postgres Advanced Server または PostgreSQL を指定します。単一のサーバータイプまたは **ALL** を指定できます。
- **History recovery** フィールドを使用して、PEM サーバーでアラートの実行の結果を保存する日数を指定します。
- **Thresholdunit** フィールドを使用して、しきい値の単位タイプを指定します。
- **Auto create** ボックスのフィールドを使用して、テンプレートを使用して自動アラートを生成するように PEM が指定します。このオプションを有効にすると、PEM は、**ターゲットタイプ**リストで指定された新しいサーバーまたはエージェントが追加されたときにアラートを作成し、ターゲットオブジェクトがドロップされたときにそのアラートを削除します。
- **Auto create?テンプレートに基づいてアラートを作成するように PEM に指定するには、**スライダーを **Yes** に設定します。
- **Operator** リストを使用して、現在のシステム値を評価するときに使用する PEM の演算子を選択します。

システム値が **しきい値**フィールドに入力された値より大きい場合にアラートをトリガーするには、大なり記号>を選択します。

小なり記号<を選択して、システム値が **しきい値**フィールドに入力された値より小さい場合にアラートをトリガーすることを示します。

- しきい値フィールドを使用して、システム値と比較する PEM の値を指定して、アラートを生成するかどうかを決定します。3 つのしきい値すべての値を指定する必要があります低、中、および高。
- **Check frequency** フィールドを使用して、アラートを実行する間のデフォルトの分数を指定します。この値は、サーバーが定義で指定された SQL コードを呼び出し、結果をテンプレートで指定されたしきい値と比較する頻度を指定します。

**Probe dependency** タブのフィールドを使用して、 **SQL** タブで指定された SQL クエリーで参照されるプローブの名前を指定します。

- **Probes** リストを使用して、使用可能なプローブのリストから選択します。
- アラートテンプレートで使用されるプローブのリストにプローブを追加するには、プローブ名を選択して **Add** を選択します。
- 削除するには選択したプローブリストからプローブを選択し、プローブ名を選択して **Delete** を選択します。

- **Parameters** タブを使用して、 **SQL** タブで指定された SQL コードで使用するパラメーターを定義します。プラス記号+を選択します。次に
- **Name** フィールドを使用して、パラメーター名を指定します。
- **Data type** リストを使用して、パラメーターのタイプを指定します。
- **Unit** フィールドを使用して、指定された単位のタイプを指定しますパラメーターによる。
- **SQL** タブの **Code** フィールドを使用して、アラートの実行時にサーバーが呼び出す SQL クエリのテキストを提供します。SQL クエリーは、しきい値と比較する結果を提供します。アラート結果が指定されたしきい値から逸脱すると、アラートが発生します。

クエリでは、変数 param_x を使用して、 **Parameters** タブで定義されたパラメーターを順番に参照します。x は、パラメーターリスト内のパラメーター定義の位置を示します。たとえば、param_1 はパラメーターリストの最初のパラメーターを参照し、param_2 はパラメーターリストの 2 番目のパラメーターを参照します。

クエリーには、次の変数を含めることもできます。

| Variable description | Variable name |
|---|---|
| agent identifier | '${agent_id}' |
| server identifier | '${server_id}' |
| database name | '${database_name}' |
| schema name | '${schema_name}' |
| Table | '${object_name}' |
| index | '${object_name}' |
| sequence | '${object_name}' |
| function name | '${object_name}' |

- **Detailed Information SQL** フィールドを使用して、アラートがトリガーされた場合に呼び出す SQL クエリーを提供します。クエリの結果セットは、詳細なアラート情報の一部として、アラートダッシュボードまたはグローバル概要ダッシュボードに表示される場合があります。

**注釈**

指定された照会が PEM 階層のさまざまなレベルサーバー、データベース、スキーマなどの 1 つ以上のプローブに依存し、プローブが無効になる場合、結果のアラートは次のように表示されます。

- アラート定義とクエリによって参照されるプローブが PEM 階層内の同じレベルからのものである場合、サーバーはグローバルアラートダッシュボードの Alert Error テーブルにアラートテンプレートを参照するアラートを表示します。 - アラート定義とクエリによって参照されるプローブが PEM 階層の異なるレベルからのものである場合、サーバーは、アラートが定義された階層の [アラートの詳細] テーブルにアラートテンプレートを参照するトリガーされたアラートを表示します。

変更を保存するには、テンプレート作成パネルで **Done** を選択します。これにより、メインのアラートテンプレートリストに戻り、**Save** を選択できます。この「一括保存」アプローチでは、1 つのセッションで複数のテンプレートを作成または編集し、ワンクリックですべての変更をコミットできます。保存すると、

カスタムアラートテンプレートは、新しいアラートを定義するための Alerting ダイアログボックスで使用できるようになります。

### アラートテンプレートのエクスポートまたはインポート

アラートテンプレートをエクスポートするには

1. **Alert Templates** タブからアラートテンプレートを選択します。
2. テーブルの右上隅にある **Export** を選択します。

これにより、アラートテンプレートが JSON ファイルとしてダウンロードフォルダーにエクスポートされます。エクスポートされた JSON ファイルを使用して、アラートテンプレートを別の PEM サーバーにインポートしたり、バックアップとして保持したりできます。

アラートテンプレートをインポートするには

1. [ **Alert Templates** ] タブで、右上隅にある **Import** を選択します。
2. コードをインポートする JSON ファイルを選択するには、 **Browse** を選択し、 **Import** を選択します。
3. インポートするファイルを選択した後、次のチェックボックスを選択できます。
    - **Skip stable** - 既に存在する場合、アラートテンプレートをスキップします。
    - **Skip 既存の依存プローブ** - アラートテンプレートはプローブに依存しています。。依存プローブが既に存在する場合、スキップするにはこのチェックボックスを選択します。

両方のチェックボックスが選択され、アラートテンプレートが既に存在する場合、アラートテンプレートのインポートはスキップされます。

**Skip stable** チェックボックスを選択せず、 **Skip dependencies** を選択し、アラートテンプレートが既に存在する場合、アラートテンプレートは正常にインポートされます。

両方のチェックボックスがオフで、アラートテンプレートが存在しない場合、アラートテンプレートは正常にインポートされます。

### アラートテンプレートの変更または削除

既存のテンプレート PEM の事前定義されたアラートテンプレートを含むの定義を表示するには、**Show System Template** リストを使用して監視対象オブジェクトのタイプを選択します。オブジェクトタイプを選択すると、そのオブジェクトタイプに対応するアラートテンプレートが Alert Templates テーブルに表示されます。

リストからテンプレート名を選択し、行の左端にある **Edit** を選択してテンプレート定義を確認します。

*Alert Templates* ダイアログボックスを使用して、アラートテンプレートに関する詳細情報を表示します。

- **General** タブには、一般的な情報が表示されます。
- **Probe dependency** タブには、テンプレートのデータを提供するプローブの名前がリストされます。
- **Parameters** タブには、SQL コードで参照されるパラメーターの名前がリストされます。
- [ **SQL** ] タブには、アラートの動作を定義する SQL コードが表示されます。

アラートテンプレートを削除するには、アラートテンプレートテーブルでテンプレート名を選択し、テーブルの右上隅にある **Delete** を選択します。アラート履歴は、テンプレート定義の **履歴保持**フィールドで指定された時間持続します。

> **注釈**
>
> 事前定義されたシステムアラートテンプレートは削除できません。削除できるのは、自分が作成したカスタムアラートテンプレートのみです。

### Predefined alert templates - reference

アラート定義には、PEM がシステム統計と比較するシステム定義またはユーザー定義の条件セットが含まれています。統計がその統計に指定された境界から逸脱すると、アラートがトリガーされ、PEM クライアントは [アラートの概要] ページに警告を表示し、オプションで監視ユーザーに通知を送信します。

以下の表は、アラートの作成に使用できるシステム定義のアラートテンプレートをリストしています。このリストは変更される場合があり、システムによって異なる場合があります。

#### エージェントに適用できるテンプレート

| Template name | Description | Probe dependency |
|---|---|---|
| Load Average (1 minute) | 1 分間のシステム負荷平均 | load_average |
| Load Average (5 minutes) | 5 分間のシステム負荷平均 | load_average |
| Load Average (15 minutes) | 15 分間のシステム負荷平均 | load_average |
| Load Average per CPU Core (1 minutes) | CPU コアごとの 1 分間のシステム負荷平均 | load_average |
| Load Average per CPU Core (5 minutes) | CPU コアごとの 5 分間のシステム負荷平均 | load_average |
| Load Average per CPU Core (15 minutes) | CPU コアごとの 15 分間のシステム負荷平均 | load_average |
| CPU utilization | 平均 CPU 使用量 | cpu_usage |
| Number of CPUs running higher than a | K%の使用率しきい値を超えて実行されている CPU の数 | cpu_usage |
| Free memory percentage | 合計システムメモリの割合としての空きメモリ | memory_usage |
| Memory used percentage | 使用されたメモリの割合 | memory_usage |
| Swap consumption | 消費されたスワップ領域メガバイト単位 | memory_usage |
| Swap consumption percentage | 消費されたスワップ領域の割合 | memory_usage |
| Disk Consumption | 消費されたディスク領域メガバイト単位 | disk_space |

次のページに続く

表 17 – 前のページからの続き

| Template name | Description | Probe dependency |
|---|---|---|
| Disk consumption percentage | 消費されたディスクの割合 | disk_space |
| Disk Available | 使用可能なディスク領域メガバイト単位 | disk_space |
| Disk busy percentage | ビジーなディスクの割合 | disk_busy_info |
| Most used disk percentage | システムで最も使用されているディスクの使用率 | disk_space |
| Total table bloat on host | ホスト上のテーブルによって浪費された合計領域 MB 単位 | table_bloat, settings |
| Highest table bloat on host | ホスト上のテーブルによって浪費された最も多くの領域 MB 単位 | table_bloat, settings |
| Average table bloat on host | ホスト上のテーブルによって浪費された平均領域 MB 単位 | table_bloat, settings |
| Table size on host | ホスト上のテーブルのサイズ MB 単位 | table_size |
| Database size on host | ホスト上のデータベースのサイズ MB 単位 | database_size |
| Number of ERRORS in the logfile on agent N in last X hours. | 過去 X 時間のエージェント N のログファイルの ERRORS の数 | N/A |
| Number of ERRORS in the audit logfile on agent N in last X hours | 過去 X 時間のエージェント N の監査ログファイルの ERRORS の数 | N/A |
| Number of WARNINGS in the logfile on agent N in last X hours | 過去 X 時間のエージェント N のログファイルの WARNING の数 | N/A |
| Number of WARNINGS in the audit logfile on agent N in last X hours | 過去 X 時間のエージェント N の監査ログファイルの WARNING の数 | N/A |
| Number of WARNINGS or ERRORS in the logfile on agent N in last X hours | 過去 X 時間のエージェント N のログファイルの WARNING または ERRORS の数 | N/A |
| Number of WARNINGS or ERRORS in audit the logfile on agent N in last X hours | 過去 X 時間のエージェント N のログファイルの WARNING または ERRORS の数 | N/A |
| Package version mismatch | カタログごとにパッケージバージョンの不一致を確認します | N/A |
| Total materialized view bloat on host | ホスト上のマテリアライズドビューによって浪費された合計領域 MB 単位 | mview_bloat, settings |
| Highest materialized view bloat on host | ホスト上のマテリアライズドビューによって無駄にされる最大の領域 MB 単位 | mview_bloat, settings |
| Average materialized view bloat on host | ホスト上のマテリアライズドビューによって浪費される平均領域 MB 単位 | mview_bloat, settings |

次のページに続く

表 17 – 前のページからの続き

| Template name | Description | Probe dependency |
|---|---|---|
| Materialized view size on host | ホスト上のマテリアライズドビューのサイズ MB 単位 | mview_size |
| Agent Down | 指定されたエージェントは現在ダウンしています | N/A |

### サーバーに適用できるテンプレート

| Template name | Description | Probe dependency |
|---|---|---|
| Audit config mismatch | 監査構成パラメーターの不一致のチェック | audit_configuration |
| Average materialized view bloat in server | サーバー内のマテリアライズドビューによって浪費される平均領域 MB 単位 | mview_bloat, settings |
| Average table bloat in server | サーバー内のテーブルによって浪費された平均領域 MB 単位 | table_bloat, settings |
| Buffers allocated per second | 最後の 2 つのプローブサイクルで、1 秒ごとに割り当てられたバッファーの数 | background_writer_statistics |
| Buffers written per second | 最後の 2 つのプローブサイクルで 1 秒ごとに書き込まれるバッファーの数 | background_writer_statistics |
| Committed transactions percentage | 過去 N 分間にコミットしたサーバー内のトランザクションとロールバックしたトランザクションの割合 | database_statistics |
| Conflicting replication slots | 特定のサーバーで競合しているスロットを確認する | |
| Connections in idle state | サーバー内のアイドル状態にある接続の数 | session_info |
| Connections in idle-in-transaction state | アイドル中トランザクション状態にあるサーバー内の接続の数 | session_info |
| Connections in idle-in-transaction state, as percentage of max_connections | サーバーで許可される最大接続の割合としての、トランザクション中アイドル状態にあるサーバー内の接続の数、設定 | session_info, settings |
| Database size in server | サーバー内のデータベースのサイズ MB 単位 | database_size |
| Dead Tuples | サーバー内の推定デッドタプルの数 | table_statistics |
| Dead tuples percentage | サーバー内の推定デッドタプルの割合 | table_statistics |
| EFM agent down | フェールオーバーマネージャーサービスがダウンしています | efm_cluster_info |
| EFM cluster failover detected | フェールオーバーは、指定された時間内に EFM クラスターのフェールオーバーイベントが記録されると、トリガーされます。 | failover_manager_node_status |
| EFM fewer than N nodes active | クラスター内のアクティブな EFM ノードの数が予想されるしきい値を下回るとトリガーされます | efm_cluster_node_status |

次のページに続く

表 18 – 前のページからの続き

| Template name | Description | Probe dependency |
|---|---|---|
| EFM missing primary | EFM クラスターにはプライマリノードがありません | efm_cluster_node_status |
| Function Count | サーバー内のファンクションの合計数 | oc_function |
| Highest materialized view bloat in server | サーバー内のマテリアライズドビューによって浪費された最も多くの領域 MB 単位 | mview_bloat, settings |
| Highest table bloat in server | サーバー内のテーブルによって無駄にされた最も多くの領域 MB 単位 | table_bloat, settings |
| Hot update percentage | 過去 N 分間のサーバー内のホットアップデートの割合 | table_statistics |
| Inactive replication slots | 特定のサーバーで非アクティブなスロットを確認する | |
| Index Scans | 過去 N 分間のサーバー内のインデックススキャンの数 | table_statistics |
| Index size as a percentage of table size | サーバー内のインデックスのサイズ、テーブルのサイズのパーセンテージ | index_size, oc_index, table_size |
| InfiniteCache buffers hit percentage | 過去 N 分間、InfiniteCache によって満たされたサーバー内のブロック読み取り要求の割合 | database_statistics |
| Largest index by table-size percentage | サーバー内の最大のインデックス、テーブルのサイズ、oc_index、table_size のパーセンテージとして計算されます。 | index_size, oc_index, table_size |
| Largest materialized view (by multiple of unbloated size) | サーバー内で最大のマテリアライズドビュー。肥大化されていない独自の推定サイズの倍数として計算されます。 N MB より小さいマテリアライズドビューを除外する | mview_bloat, settings |
| Largest table (by multiple of unbloated size) | サーバー内で最大のテーブル。肥大化されていない独自の推定サイズの倍数として計算されます。 N MB より小さいテーブルを除外する | table_bloat, settings |
| Last Analyze | サーバーでの最後の分析からの時間 | table_statistics |
| Last AutoAnalyze | サーバーでの最後の自動分析からの時間 | table_statistics |
| Last AutoVacuum | サーバーでの最後の自動バキュームからの時間 | table_statistics |
| Last Vacuum | サーバーでの最後のバキュームからの時間 | table_statistics |
| Live Tuples | サーバー内の推定ライブタプルの数 | table_statistics |
| Log config mismatch | ログ構成パラメーターの不一致のチェック | log_configuration |
| Long-running autovacuums | N 秒以上実行されているサーバー内の自動バキューム操作の数 | session_info |
| Long-running idle connections | N 秒以上アイドル状態になったサーバー内の接続の数 | session_info |
| Long-running idle connections and idle transactions | N 秒を超えてアイドル状態になったサーバー内の接続、またはトランザクション内アイドル状態のトランザクションの数 | session_info |
| Long-running idle transactions | トランザクションで N 秒を超えてアイドル状態になったサーバー内の接続の数 | session_info |

次のページに続く

表 18 – 前のページからの続き

| Template name | Description | Probe dependency |
|---|---|---|
| Long-running queries | N 秒以上実行されているサーバー内のクエリーの数 | session_info |
| Long-running transactions | N 秒以上実行されているサーバー内のトランザクションの数 | session_info |
| Long-running vacuums | N 秒以上実行されているサーバー内のバキューム操作の数 | session_info |
| Materialized View Count | サーバー内のマテリアライズドビューの合計数 | oc_views |
| Materialized view size in server | サーバー内のマテリアライズドビューのサイズ MB 単位 | mview_size |
| Multixact ID exhaustion (wraparound) | multixact ID の枯渇の確認ラップアラウンド | |
| Number of attacks detected in the last N minutes | 過去 N 分間に発生した SQL インジェクション攻撃の数 | sql_protect |
| Number of attacks detected in the last N minutes by username | 過去 N 分間に発生したユーザー名別の SQL インジェクション攻撃の数 | sql_protect |
| Number of ERRORS in the logfile on server M in the last X hours | 過去 X 時間のサーバー M のログファイルの ERRORS の数 | N/A |
| Number of minutes lag of replica server from primary server | ストリーミングレプリケーション レプリカノードがプライマリノードから遅れている分数 | streaming_replication_lag_time |
| Number of prepared transactions | 準備済み状態のトランザクションの数 | number_of_prepared_transactions |
| Number of replica servers lag behind the primary by flush location | ストリーミングレプリケーション レプリカサーバーの数は、フラッシュロケーションごとにプライマリよりも遅れています | streaming_replication |
| Number of replica servers lag behind the primary by replay location | ストリーミングレプリケーション レプリカサーバーの数は、再生場所ごとにプライマリよりも遅れています | streaming_replication |
| Number of replica servers lag behind the primary by write location | ストリーミングレプリケーション レプリカサーバーの数は、書き込み場所ごとにプライマリよりも遅れています | streaming_replication |
| Number of users expiring in N days | アカウントの有効期限が N 日になるユーザーの数 | user_info |
| Number of users whose password expiring in N days | パスワードの有効期限が切れた、または N 日で有効期限が切れるユーザーの数 | user_info |

次のページに続く

表 18 – 前のページからの続き

| Template name | Description | Probe dependency |
|---|---|---|
| Number of WAL archives pending | ストリーミングレプリケーション レプリカで再生保留中の WAL ファイルの数 | wal_archive_status |
| Number of WAL files | 先行書き込みログファイルの合計数 | number_of_wal_files |
| Number of WARNINGS in the logfile on server M in the last X hours | 過去 X 時間のサーバー M のログファイルの WARNING の数 | N/A |
| Number of WARNINGS or ERRORS in the logfile on server M in the last X hours | 過去 X 時間のサーバー M のログファイルの WARNING または ERRORS の数 | N/A |
| Patroni cluster paused | Patroni Patroni クラスターが一時停止され、フェールオーバーに使用できないかどうかを検出します。 | patroni_cluster_status |
| Patroni cluster failover detected | Patroni フェールオーバーは、指定された時間内に Patroni クラスターのフェールオーバーイベントが記録されると、トリガーされます。 | patroni_node_status |
| Patroni DCS not healthy | Patroni 分散構成ストア etcd が正常でないかどうかを検出します。 | patroni_cluster_status |
| Patroni down or out of contact | Patroni Patroni プロセスが到達不能であるか、監視対象ノードで障害が発生しているかどうかを検出します。 | patroni_cluster_status |
| Patroni no leader detected | Patroni Patroni クラスターにリーダー/マスターノードがない場合を検出します。 | patroni_cluster_status |
| Patroni timeline mismatch | Patroni ノードのタイムラインがクラスタータイムラインと一致しないかどうかを検出します。 | patroni_node_status, patroni_cluster_status |
| Percentage of buffers written by backends | バックエンドによって書き込まれたバッファーの割合と書き込まれた合計バッファーの割合 | background_writer_statistics |
| Percentage of buffers written by backends over the last N minutes | バックエンドによって書き込まれたバッファの割合と最後の N のバッファバックエンドの合計 | background_writer_statistics |
| Percentage of buffers written by checkpoint | チェックポイントによって書き込まれたバッファーの割合と書き込まれた合計バッファーの割合 | background_writer_statistics |
| PGD Group Raft Consensus | PGD グループ Raft コンセンサスが機能しない | bdr_monitor_group_raft |
| PGD Group Raft Leader ID not matching | PGD グループの Raft リーダー ID が一致しません | bdr_group_raft_details |
| PGD Group versions check | PGD グループ内の PGD/pg 論理バージョンが不一致です | bdr_monitor_group_raft |
| PGD worker error detected | PGD ノードで検出された PGD ワーカーエラーが報告されました | |

次のページに続く

表 18 – 前のページからの続き

| Template name | Description | Probe dependency |
|---|---|---|
| Replica server lag behind the primary by flush location | ストリーミングレプリケーション レプリカサーバーは、MB 単位のフラッシュロケーションにより、プライマリよりも遅延します。 | streaming_replication |
| Replica server lag behind the primary by replay location | ストリーミングレプリケーション レプリカサーバーは、MB 単位の再生場所によりプライマリよりも遅延します。 | streaming_replication |
| Replica server lag behind the primary by size (MB) | ストリーミングレプリケーション レプリカサーバーは MB 単位のサイズでプライマリよりも遅延しています | streaming_replication |
| Replica server lag behind the primary by WAL pages | ストリーミングレプリケーション レプリカサーバーは WAL ページごとにプライマリよりも遅延します | streaming_replication |
| Replica server lag behind the primary by WAL segments | ストリーミングレプリケーション レプリカサーバーは WAL セグメントごとにプライマリよりも遅延しています | streaming_replication |
| Replica server lag behind the primary by write location | ストリーミングレプリケーション レプリカサーバーは、MB 単位の書き込み場所によりプライマリよりも遅延します。 | streaming_replication |
| Sequence Count | サーバー内のシーケンスの合計数 | oc_sequence |
| Sequential Scans | 過去 N 分間のサーバー内のフルテーブルスキャンの数 | table_statistics |
| Server Down | 指定されたサーバーは現在アクセスできません | N/A |
| Shared buffers hit percentage | 過去 N 分間、共有バッファによって満たされたサーバー内のブロック読み取り要求の割合 | database_statistics |
| Table Count | サーバー内のテーブルの合計数 | oc_table |
| Table size in server | サーバー内のテーブルのサイズ MB 単位 | table_size |
| Tablespace directory missing | テーブルスペースディレクトリが欠落しているか、オペレーティングシステムから削除されている場合に、アラートをトリガーします。 | tablespace_size |
| Total connections | サーバー内の合計接続数 | session_info |
| Total connections as percentage of max_connections | サーバーで許可される最大接続の割合としてのサーバー内の合計接続数、設定 | session_info, settings |
| Total materialized view bloat in server | サーバー内のマテリアライズドビューによって浪費された合計領域 MB 単位 | mview_bloat, settings |
| Total table bloat in server | サーバー内のテーブルによって浪費された合計領域 MB 単位 | table_bloat, settings |
| Transaction ID exhaustion (wraparound) | トランザクション ID の枯渇の確認ラップアラウンド | |
| Tuples deleted | 過去 N 分間にサーバーから削除されたタプル | database_statistics |
| Tuples fetched | 過去 N 分間にサーバーから取得されたタプル | database_statistics |
| Tuples hot updated | 過去 N 分間にサーバーでタプルがホット更新されました | table_statistics |

次のページに続く

表 18 – 前のページからの続き

| Template name | Description | Probe dependency |
|---|---|---|
| Tuples inserted | 過去 N 分間にサーバーに挿入されたタプル | database_statistics |
| Tuples returned | 過去 N 分間にサーバーから返されたタプル | database_statistics |
| Tuples updated | 過去 N 分間にサーバーで更新されたタプル | database_statistics |
| Ungranted locks | サーバー内の未付与のロックの数 | blocked_session_info |
| Unused, non-superuser connections | サーバー上の未使用の非スーパーユーザー接続の数、user_info、設定 | session_info, user_info, settings |
| Unused, non-superuser connections as percentage of max_connections | max_connections、user_info、settings の max_connections の割合としてのサーバー上の未使用の非スーパーユーザー接続の数 | session_info, user_info, settings |
| View Count | サーバー内のビューの合計数 | oc_views |

### データベースに適用できるテンプレート

| Template name | Description | Probe dependency |
|---|---|---|
| Total table bloat in database | データベース内のテーブルによって浪費された合計領域 MB 単位 | table_bloat, settings |
| Largest table (by multiple of unbloated size) | データベース内で最大のテーブル。肥大化していない独自の推定サイズの倍数として計算されます。 N MB より小さいテーブルを除外する | table_bloat, settings |
| Highest table bloat in database | データベース内のテーブルによって浪費された最も多くの領域 MB 単位 | table_bloat, settings |
| Average table bloat in database | データベース内のテーブルによって浪費された平均領域 MB 単位 | table_bloat, settings |
| Table size in database | データベース内のテーブルのサイズ MB 単位 | table_size |
| Database size | データベースのサイズ MB 単位 | database_size |
| Total connections | データベース内の合計接続数 | session_info |
| Total connections as percentage of max_connections | サーバーで許可される最大接続の割合としてのデータベース内の接続の合計数、設定 | session_info, settings |
| Ungranted locks | データベース内の未付与のロックの数 | blocked_session_info |
| Connections in idle state | データベース内のアイドル状態にある接続の数 | session_info |
| Connections in idle-in-transaction state | データベース内のアイドルイントランザクション状態にある接続の数 | session_info |
| Connections in idle-in-transaction state,as percentage of max_connections | サーバーで許可される最大接続の割合としての、トランザクション中アイドル状態にあるデータベース内の接続の数、設定 | session_info, settings |
| Long-running idle connections | N 秒以上アイドル状態であるデータベース内の接続の数 | session_info |

次のページに続く

表 19 – 前のページからの続き

| Template name | Description | Probe dependency |
|---|---|---|
| Long-running idle connections and idle transactions | N 秒を超えてアイドルまたはトランザクション中アイドル状態であったデータベース内の接続の数 | session_info |
| Long-running idle transactions | トランザクションで N 秒を超えてアイドル状態になったデータベース内の接続の数 | session_info |
| Long-running transactions | N 秒以上実行されているデータベース内のトランザクションの数 | session_info |
| Long-running queries | N 秒以上実行されているデータベース内のクエリーの数 | session_info |
| Long-running vacuums | N 秒を超えて実行されたデータベースのバキューム操作の数 | session_info |
| Long-running autovacuums | N 秒以上実行されているデータベースの自動バキューム操作の数 | session_info |
| Committed transactions percentage | 過去 N 分間にコミットされたデータベース内のトランザクションとロールバックされたトランザクションの割合 | database_statistics |
| Shared buffers hit percentage | 過去 N 分間、共有バッファによって満たされたデータベース内のブロック読み取り要求の割合 | database_statistics |
| InfiniteCache buffers hit percentage | 過去 N 分間、InfiniteCache によって満たされたデータベース内のブロック読み取り要求の割合 | database_statistics |
| Tuples fetched | 過去 N 分間にデータベースから取得されたタプル | database_statistics |
| Tuples returned | 過去 N 分間にデータベースから返されたタプル | database_statistics |
| Tuples inserted | 過去 N 分間にデータベースに挿入されたタプル | database_statistics |
| Tuples updated | 過去 N 分間にデータベースで更新されたタプル | database_statistics |
| Tuples deleted | 過去 N 分間にデータベースから削除されたタプル | database_statistics |
| Tuples hot updated | 過去 N 分間にデータベースのタプルがホット更新されました | table_statistics |
| Sequential Scans | 過去 N 分間のデータベースのフルテーブルスキャンの数 | table_statistics |
| Index Scans | 過去 N 分間のデータベースのインデックススキャンの数 | table_statistics |
| Hot update percentage | 過去 N 分間のデータベースのホット更新の割合 | table_statistics |
| Live Tuples | データベース内の推定ライブタプルの数 | table_statistics |
| Dead Tuples | データベース内の推定デッドタプルの数 | table_statistics |
| Dead tuples percentage | データベース内の推定デッドタプルの割合 | table_statistics |
| Last Vacuum | データベースの最後のバキュームからの時間 | table_statistics |
| Last AutoVacuum | データベースの最後の自動バキュームからの時間 | table_statistics |
| Last Analyze | データベースの最後の分析からの時間 | table_statistics |
| Last AutoAnalyze | データベースの最後の自動分析からの時間 | table_statistics |
| Table Count | データベース内のテーブルの合計数 | oc_table |

次のページに続く

表 19 – 前のページからの続き

| Template name | Description | Probe dependency |
|---|---|---|
| Function Count | データベース内のファンクションの合計数 | oc_function |
| Sequence Count | データベース内の配列の合計数 | oc_sequence |
| Index size as a percentage of table size | データベース内のインデックスのサイズ、テーブルのサイズのパーセンテージ | table_size |
| Largest index by table-size percentage | テーブルのサイズ、oc_index、table_size のパーセンテージとして計算されたデータベース内の最大のインデックス | index_size, oc_index, table_size |
| Database Frozen XID | データベースの凍結されたトランザクション ID の経過時間現在のトランザクションの前のトランザクション | database_frozenxid |
| Number of attacks detected in the last N minutes | 過去 N 分間に発生した SQL インジェクション攻撃の数 | sql_protect |
| Number of attacks detected in the last N minutes by username | 過去 N 分ごとにユーザー名ごとに発生した SQL インジェクション攻撃の数 | sql_protect |
| Queries that have been cancelled due to dropped tablespaces | ストリーミングレプリケーション テーブルスペースが削除されたためにキャンセルされたクエリの数 | streaming_replication_db_conflicts |
| Queries that have been cancelled due to lock timeouts | ストリーミングレプリケーション ロックタイムアウトのためにキャンセルされたクエリの数 | streaming_replication_db_conflicts |
| Queries that have been cancelled due to old snapshots | ストリーミングレプリケーション 古いスナップショットが原因でキャンセルされたクエリの数 | streaming_replication_db_conflicts |
| Queries that have been cancelled due to pinned buffers | ストリーミングレプリケーション 固定されたバッファのためキャンセルされたクエリの数 | streaming_replication_db_conflicts |
| Queries that have been cancelled due to deadlocks | ストリーミングレプリケーション デッドロックのためにキャンセルされたクエリの数 | streaming_replication_db_conflicts |
| Total events lagging in all slony clusters | Slony レプリケーション すべての Slony クラスターで遅延しているイベントの合計 | slony_cluster |
| Events lagging in one slony cluster | Slony レプリケーション 1 つのスロニークラスターで遅延しているイベント | slony_cluster |
| Lag time (minutes) in one slony cluster | Slony レプリケーション 1 つの slony クラスターでの遅延時間分 | slony_cluster |
| Total rows lagging in xdb single primary replication | xDB レプリケーション xdb シングルプライマリレプリケーションの合計行遅延 | xdb_smr_mmr_replication |
| Total rows lagging in xdb multi primary replication | xDB レプリケーション xdb マルチプライマリレプリケーションの合計行遅延 | xdb_smr_mmr_replication |

次のページに続く

表 19 – 前のページからの続き

| Template name | Description | Probe dependency |
|---|---|---|
| Total materialized view bloat in database | データベース内のマテリアライズドビューによって浪費された合計領域 MB 単位 | mview_bloat, settings |
| Largest materialized view (by multiple of unbloated size) | データベース内の最大のマテリアライズドビュー。肥大化されていない推定サイズの倍数として計算されます。 N MB より小さいマテリアライズドビューを除外する | mview_bloat, settings |
| Highest materialized view bloat in database | データベース内のマテリアライズドビューによって浪費された最も多くの領域 MB 単位 | mview_bloat, settings |
| Average materialized view bloat in database | データベース内のマテリアライズドビューによって浪費された平均領域 MB 単位 | mview_bloat, settings |
| Materialized view size in database | データベース内のマテリアライズドビューのサイズ MB 単位 | mview_size |
| View Count | データベース内のビューの合計数 | oc_views |
| Materialized View Count | データベース内のマテリアライズドビューの合計数 | oc_views |

### スキーマに適用できるテンプレート

| Template name | Description | Probe dependency |
|---|---|---|
| Total table bloat in schema | スキーマ内のテーブルによって浪費された合計領域 MB 単位 | table_bloat, settings |
| Largest table (by multiple of unbloated size) | スキーマ内で最大のテーブル。肥大化されていない独自の推定サイズの倍数として計算されます。 N MB より小さいテーブルを除外する | table_bloat, settings |
| Highest table bloat in schema | スキーマ内のテーブルによって浪費された最大領域 MB 単位 | table_bloat, settings |
| Average table bloat in schema | スキーマ内のテーブルによって浪費された平均領域 MB 単位 | table_bloat, settings |
| Table size in schema | スキーマ内のテーブルのサイズ MB 単位 | table_size |
| Tuples inserted | 過去 N 分間にスキーマに挿入されたタプル | table_statistics |
| Tuples updated | 過去 N 分間にスキーマで更新されたタプル | table_statistics |
| Tuples deleted | 過去 N 分間にスキーマから削除されたタプル | table_statistics |
| Tuples hot updated | 過去 N 分間にスキーマのタプルがホット更新されました | table_statistics |
| Sequential Scans | 過去 N 分間のスキーマのフルテーブルスキャンの数 | table_statistics |
| Index Scans | 過去 N 分間のスキーマのインデックススキャンの数 | table_statistics |
| Hot update percentage | 過去 N 分間のスキーマのホット更新の割合 | table_statistics |
| Live Tuples | スキーマ内の推定ライブタプルの数 | table_statistics |
| Dead Tuples | スキーマ内の推定デッドタプルの数 | table_statistics |
| Dead tuples percentage | スキーマ内の推定デッドタプルの割合 | table_statistics |

次のページに続く

表 20 – 前のページからの続き

| Template name | Description | Probe dependency |
|---|---|---|
| Last Vacuum | スキーマの最後のバキュームからの時間 | table_statistics |
| Last AutoVacuum | スキーマの最後の自動バキュームからの時間 | table_statistics |
| Last Analyze | スキーマの最後の分析からの時間 | table_statistics |
| Last AutoAnalyze | スキーマの最後の自動分析からの時間 | table_statistics |
| Table Count | スキーマ内のテーブルの合計数 | oc_table |
| Function Count | スキーマ内のファンクションの合計数 | oc_function |
| Sequence Count | スキーマ内のシーケンスの合計数 | oc_sequence |
| Index size as a percentage of table size | スキーマ内のインデックスのサイズ、テーブルのサイズのパーセンテージ | table_size |
| Largest index by table-size percentage | テーブルのサイズ、oc_index、table_size のパーセンテージとして計算されるスキーマ内の最大のインデックス | index_size, oc_index, table_size |
| Materialized view bloat | マテリアライズドビューによって無駄にされた領域 MB 単位 | mview_bloat, settings |
| Total materialized view bloat in schema | スキーマのマテリアライズドビューによって浪費された合計領域 MB 単位 | mview_bloat, settings |
| Materialized view size as a multiple of unbloated size | 推定される肥大化されていないサイズの倍数としてのマテリアライズドビューのサイズ | mview_bloat |
| Largest materialized view (by multiple of unbloated size) | スキーマ内の最大のマテリアライズドビュー。肥大化されていない独自の推定サイズの倍数として計算されます。 N MB より小さいマテリアライズドビューを除外する | mview_bloat, settings |
| Highest materialized view bloat in schema | スキーマのマテリアライズドビューによって浪費される最大領域 MB 単位 | mview_bloat, settings |
| Average materialized view bloat in schema | スキーマのマテリアライズドビューによって浪費される平均領域 MB 単位 | mview_bloat, settings |
| Materialized view size | マテリアライズドビューのサイズ MB 単位 | mview_size |
| Materialized view size in schema | スキーマのマテリアライズドビューのサイズ MB 単位 | mview_size |
| View Count | スキーマ内のビューの合計数 | oc_views |
| Materialized View Count | スキーマ内のマテリアライズドビューの合計数 | ov_views |
| Materialized View Frozen XID | マテリアライズドビューの凍結されたトランザクション ID の経過時間現在のトランザクションの前のトランザクション | mview_frozenxid |

### テーブルに適用できるテンプレート

| Template name | Description | Probe dependency |
|---|---|---|
| Table bloat | テーブルによって無駄にされた領域 MB 単位 | table_bloat, settings |
| Table size | テーブルのサイズ MB 単位 | table_size |

次のページに続く

表 21 – 前のページからの続き

| Template name | Description | Probe dependency |
|---|---|---|
| Table size as a multiple of unbloated size | 推定される肥大化されていないサイズの倍数としてのテーブルのサイズ | table_bloat |
| Tuples inserted | 過去 N 分間にテーブルに挿入されたタプル | table_statistics |
| Tuples updated | 過去 N 分間にテーブルで更新されたタプル | table_statistics |
| Tuples deleted | 過去 N 分間にテーブルから削除されたタプル | table_statistics |
| Tuples hot updated | 過去 N 分間にテーブル内のタプルがホット更新されました | table_statistics |
| Sequential Scans | 過去 N 分間のテーブルのフルテーブルスキャンの数 | table_statistics |
| Index Scans | 過去 N 分間のテーブルのインデックススキャンの数 | table_statistics |
| Hot update percentage | 過去 N 分間のテーブル内のホット更新の割合 | table_statistics |
| Live Tuples | テーブル内の推定ライブタプルの数 | table_statistics |
| Dead Tuples | テーブル内の推定デッドタプルの数 | table_statistics |
| Dead tuples percentage | テーブル内の推定デッドタプルの割合 | table_statistics |
| Last Vacuum | テーブル上の最後のバキュームからの時間 | table_statistics |
| Last AutoVacuum | テーブルの最後の自動バキュームからの時間 | table_statistics |
| Last Analyze | テーブルでの最後の分析からの時間 | table_statistics |
| Last AutoAnalyze | テーブルでの最後の自動分析からの時間 | table_statistics |
| Row Count | テーブルの推定行数 | table_statistics |
| Index size as a percentage of table size | テーブルのインデックスのサイズ、テーブルのサイズのパーセンテージ | table_size |
| Table Frozen XID | テーブルの凍結されたトランザクション ID の経過時間現在のトランザクションの前のトランザクション | table_frozenxid |

### グローバルテンプレート

| Template name | Description | Probe dependency |
|---|---|---|
| Agents Down | 最近報告されていないエージェントの数 | N/A |
| Servers Down | 現在アクセスできないサーバーの数 | N/A |
| Alert Errors | エラー状態のアラートの数 | N/A |

### Defining a new alert

PEM クライアントの **Manage Alerts** タブを使用して、アラートを定義、コピー、または管理します。 [ **Manage Alerts** ] タブを開くには、 **Management > Alerts > Manage Alerts** を選択します。

[ **Manage Alerts** ] タブには、PEM クライアントツリーで現在選択されているオブジェクトに定義されているアラートのテーブルが表示されます。Alerts テーブルを使用して、既存のアラートを変更したり、新しいアラートを作成したりできます。

アラートエディターを開き、アラートを作成するには、テーブルの右上にあるプラス記号+を選択します。

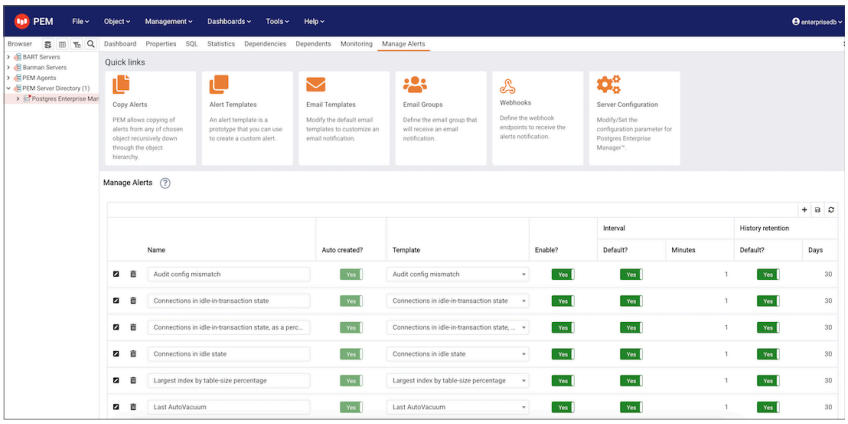



図 7: The Manage Alerts tab

エディターが開きます。

**General** タブのフィールドを使用して、アラートに関する情報を提供します。

- **Name** フィールドにアラートの名前を入力します。
- **Template** リストを使用して、アラートのテンプレートを選択します。アラートテンプレートは、1 つ以上のメトリックまたはパラメーターを使用して、PEM がユーザー指定のアラート境界を比較する値を生成するファンクションです。テンプレートファンクションによって返された値が、**演算子** および **しきい値**フィールドで指定されたユーザー定義のアラートの境界内の値と評価された場合、PEM:
- アラートを発生させます b_tran_4 通知を追加します表示されるアラートの概要 b_tran_4 テンプレートで指定されたアクションを実行します
- **Enable?** スイッチを使用して、アラートを有効にする **Yes** か、無効にする **No** を指定します。
- **Interval** ボックスを使用して、アラート条件が満たされたかどうかをアラートが確認する頻度を指定します。**Minutes** セレクターを使用して、間隔値を指定します。 **Default** スイッチを使用して、**Minutes** 値を選択したテンプレートのデフォルト推奨値に設定またはリセットします。
- **History recovery** ボックスを使用して、アラートによって収集されたデータを PEM が保存する日数を指定します。**Days** セレクターを使用して、データを保存する日数を指定します。**Default** スイッチを使用して、 **Days** 値をデフォルト値 30 日に設定またはリセットします。
- **しきい値**ボックスのコントロールを使用して、アラートのトリガー基準を定義します。**しきい値**フィールドで指定された値がシステム値演算子で指定より大きいまたは小さいと評価されると、PEM は低、中、または高のアラートレベルを引き上げます。
- **Operator** リストを使用して、現在のシステム値を評価するときに使用する PEM の演算子を選択します。
- システム値が入力された値より大きい場合にアラートをトリガーするには、大なり記号>を選択します**しきい値** フィールド.
- システム値が **しきい値**フィールドに入力された値より小さい場合にアラートをトリガーするには、小なり記号<を選択します。
- **Threshold** フィールドを使用して、システム値と比較する PEM の値を指定して、アラートを生成するかどうかを決定します。3 つのしきい値すべての値を指定する必要があります低、中、および高。

Parameter Options テーブルには、選択したテンプレートに必要なパラメーターのリストが含まれています。テーブルには、事前定義されたパラメーターと値を指定する必要があるパラメーターの両方が表示されます。**Value** 列にプロンプトを表示するパラメーターの値を指定する必要があります。

PEM は、アラートがトリガーされた場合、またはアラートがクリアされた場合に、通知を送信したり、スクリプトを実行したりできます。**Notification** タブを使用して、アラートが生成された場合に PEM がどのように動作するかを指定します。

**電子メール通知**ボックスを使用して、指定したレベルでアラートがトリガーされた場合に電子メール通知を受信する電子メールグループを指定します。*Email Groups* タブを使用して、アラートがトリガーされたときに通知するユーザーのアドレスを含むメールグループを作成します。[ **Send the notifications** ] タブにアクセスするには、[ **Manage Alerts** ] タブの [ **Quick Links** ] メニューにある **Email Groups** を選択します。

- 特定のアラートレベルに到達したときに電子メールを送信するように PEM に指示するには、アラートレベルの横にあるスライダーを **Yes** に設定します。リストを使用して、通知する事前定義されたユーザーまたはグループを選択します。

PEM が電子メール通知を送信する前に、SMTP サーバーを使用して電子メールを配信するように PEM サーバーを構成する必要があります。

指定したレベルでアラートがトリガーされる場合、 **Webhook notification** ボックスを使用して 1 つ以上のエンドポイントを指定します。*webhooks tab* を使用して、アラートがトリガーされたときに通知を受信する Webhook エンドポイントを作成します。[ **Webhooks** ] タブにアクセスするには、[ **Manage Alerts** ] タブの [ **Quick Links** ] メニューにある **Webhooks** を選択します。

- アラート通知を Webhook エンドポイントに送信するには、 **Enable?** を **Yes** に設定します。
- 要件に従ってカスタマイズされたアラートレベルを設定するには、**Overridedefault configuration?** を **Yes** に設定します。 **Yes** に設定すると、すべてのアラートレベルの構成が有効になります。
- リストを使用して、**低アラート?** 、**中程度のアラート?** 、**高アラート?** 、および **クリアされたアラート?** の通知を送信する事前定義されたエンドポイントを選択します。

**Trap notification** オプションを使用して、このアラートのトラップ通知を構成します。

- このアラートの状態が変更されたときに SNMP トラップ通知を送信するには、**Send attempts** を **Yes** に設定します。
- SNMP バージョンを識別するには、 **SNMP Ver** を **v1** 、 **v2** 、または **v3** に設定します。
- **低アラート** 、 **中アラート** 、および **高アラート**スライダーを使用して、トラップをトリガーするアラートレベルを選択します。たとえば、**高アラート** の横にあるスライダーを **Yes** に設定すると、高重大度レベルのアラートがトリガーされたときに PEM は通知を送信します。

通知を送信する前に、SNMP トラップ/通知レシーバーに通知を送信するように PEM サーバーを構成する必要があります。SNMP v3 トラップを送信するために、PEM エージェントは SNMP パケットの認証、暗号化、および復号化を担当する’User Security Model(USM)’を使用します。

SNMP v3 トラップの送信中に、エージェントは `snmp_boot_counter` ファイルを作成します。このファイルは、 `agent.cfg` の `batch_script_dir` パラメーターで指定された場所に作成されます。このパラメーターが構成されていない場合、または認証制限のためディレクトリにアクセスできない場合、ファイルはオペレーティングシステムの一時ディレクトリに作成されます。それも不可能な場合、ファイルはホームディレクトリに作成されます。

**Nagios notification** ボックスを使用して、アラートがトリガーまたはクリアされたときに Nagios ネットワークアラートソフトウェアに通知するように PEM サーバーに指示します。詳細はこちら *Using PEM with Nagios*

- アラートがトリガーまたはクリアされたときに Nagios に通知するには、**Submit Passive service check result to Nagios** スイッチを **Yes** に設定します。
- **Script attempts** ボックスを使用して、アラートがトリガーされた場合に実行するスクリプトをオプションで定義し、スクリプトの実行に関する詳細を指定します。
- **Execute script** スライダーを **Yes** に設定して、アラートがトリガーされた場合に提供されたスクリプトを実行するように PEM に指示します。
- **Execute on agent individually** スライダーを **Yes** に設定して、アラートをトリガーした状況が解決されたときに提供されたスクリプトを実行するように PEM に指示します。
- **Execute script on** オプションを使用して、PEM サーバーまたは監視対象サーバーで実行するスクリプトを示します。
- **Code** フィールドに、実行する PEM のスクリプトを入力します。バッチ/シェルスクリプトを提供できます。スクリプトでは、次のプレースホルダーを使用できます。

`%AlertName%` - トリガーされたアラートの名前。

`%ObjectName%` - アラートがトリガーされたサーバーまたはエージェントの名前。

`%ThresholdValue%` - アラートがトリガーされたときにメトリックが到達したしきい値。

`%CurrentValue%` - アラートをトリガーしたメトリックの現在値。

`%CurrentState%` - アラートの現在の状態。

`%OldState%` - アラートの以前の状態。

`%AlertRaisedTime%` - アラートが発生した時間、またはアラート状態が変更された最新の時間。

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` パラメーターのエントリーを変更し、スクリプトの実行に使用するユーザーを指定する必要があります。このパラメーターには、非 root ユーザーまたは root を指定できます。ユーザーを指定しない場合、または指定され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\EnterpriseDB\PEM\agent` にあります。

アラート属性を定義したら、 **Edit** を選択してアラート定義エディターを閉じ、Alerts テーブルの右上隅にある **Save** を選択します。

変更を破棄するには、 **Refresh** を選択します。変更を破棄することを確認するメッセージが表示されます。

**注釈**

外部スクリプトでアラート構成プレースホルダー値を使用する必要があるとします。これは、コマンドライン引数として渡すか、環境変数としてエクスポートすることにより実行できます。外部スクリプトには、適切な実行権限が必要です。

- プレースホルダーをコマンドライン引数として使用してスクリプトを実行できます。

例

```
#!/bin/bash

bash <path_to_script>/script.sh "%AlertName%  %AlertLevel% %AlertDetails%"
```

- プレースホルダーの環境変数を定義し、スクリプトでそれらの環境変数を使用できます。

例

```
#!/bin/bash

export AlertName=%AlertName%
export AlertState=%AlertState%

bash <path_to_script>/script.sh
```

### Modifying an alert

Alerts テーブルを使用して、既存のアラートを管理するか、新しいアラートを作成します。PEM クライアントツリーでオブジェクトを選択して、そのオブジェクトを監視するアラートを表示します。

Alerts テーブルでアラートの一部のプロパティを変更できます。

- [Alert name] 列には、アラートの名前が表示されます。アラート名を変更するには、テーブル内の名前を置き換えて、**Save** を選択します。
- [アラートテンプレート] 列には、アラートで使用されるプロパティを指定するアラートテンプレートの名前が表示されます。リストを使用して、アラートに関連付けられているアラートテンプレートを変更できます。
- **Alert allow?** スイッチを使用して、アラートを有効にする **Yes** か、無効にする **No** かを指定します。
- Interval 列を使用して、PEM がアラート条件が満たされるかどうかをチェックする頻度を指定します。**Default** スイッチを **No** に設定し、代替値を分単位で指定します。または、 **Default** スイッチを **Yes** に設定して、値をデフォルト設定にリセットします。デフォルトでは、PEM は 1 分ごとに各アラートのステータスを確認します。
- **History recovery** フィールドを使用して、アラートによって収集されたデータを PEM が保存する日数を指定します。**Default** スイッチを **No** に設定し、日単位の代替値を指定します。または、 **Default** スイッチを **Yes** に設定して、値をデフォルト設定にリセットします。デフォルトでは、PEM は 30 日間の履歴データを保存します。

アラートを変更した後、テーブルの右上隅にある **Save** を選択して変更を保存します。

他のアラート属性を変更するには、アラート名の左側にある **Edit** を選択してエディターを開きます。エディターは、完全なアラート定義へのアクセスを提供します。

[ Alert Details ] ダイアログボックスを使用して、選択したアラートの定義を変更します。アラート定義を変更した後、[ **Save** ] を選択します。

### Deleting an alert

削除するアラートをマークするには、アラート名の横にある **Delete** ボタンを選択し、 **Yes** をクリックして確認します。これにより、メインのアラートテンプレートリストに戻ります。

このプロセスを繰り返して、複数のアラートを削除対象にマークできます。

これらの削除を永続的に適用するには、右上隅の **Save** を選択します。この「一括保存」アプローチでは、複数の変更をキューに入れ、ワンクリックでそれらをすべてコミットできます。

### Copying an alert

PEM システムでのアラートの展開を高速化するために、1 つのオブジェクトから 1 つ以上のターゲットオブジェクトにアラート定義をコピーできます。

オブジェクトからアラートをコピーするには、メイン PEM ウィンドウの PEM クライアントツリーでオブジェクトを選択します。次に、**Management > Alerts > Copy Alerts** を選択します。 [ **Manage Alerts** ] タブで、[Quick Links] ツールバーから [ **Copy Alerts** ] を選択します。

Copy Alert Configuration ダイアログボックスは、PEM クライアントツリーで選択したオブジェクトからダイアログボックスで選択したオブジェクトにすべてのアラートをコピーします。ツリーを展開して、ターゲットオブジェクトとして指定するノードを選択します。ツリーには、ソースオブジェクトの横に赤い警告インジケータが表示されます。

アラートを一度に複数のオブジェクトにコピーするには、ターゲットの親ノードを選択します。たとえば、1 つのテーブルからスキーマ内のすべてのテーブルにアラートをコピーするには、スキーマの横にあるチェックボックスをオンにします。PEM は、ソースオブジェクトと同じタイプのターゲットにのみアラートをコピーします。

PEM がコピーされるものと同じ名前のターゲットオブジェクトの既存のアラートを更新しないようにするには、**Ignore Duplications** を選択します。

既存のアラートをソースオブジェクトからの同じ名前のアラートに置き換えるには、**Replace promotes** を選択します。

**Delete Existing Alerts** を選択して、ターゲットオブジェクトからすべてのアラートを削除し、すべてのアラートをソースオブジェクトからターゲットオブジェクトにコピーします。

**Configure Alerts** を選択して、ソースオブジェクトから Copy Alert Configuration ダイアログボックスで選択したオブジェクトまたはその下にある同じタイプのすべてのオブジェクトにアラートをコピーします。

### アラートブラックアウトをスケジュールする

メンテナンス中にサーバーまたはエージェントのアラートブラックアウトをスケジュールするには、アラートブラックアウト管理 を参照してください。

### Alert blackout

アラートブラックアウトは、特定のサーバーまたはエージェントのアラート評価と通知を一時的に抑制します。停電中、PEM は通常に監視データの収集を継続し、アラートと通知のみが抑制されます。ブラックアウトは、計画的なメンテナンス時間枠またはインフラストラクチャの移行中に、一時的な中断が誤検出アラートをトリガーする場合に役立ちます。

#### ブラックアウトソース

すべてのブラックアウトには、それがどのように作成されたかを示すソースがあります。

| Source | Description | Created by |
|---|---|---|
| *manual* | 終了時間が定義されていない即時オンデマンドのブラックアウト | Alert Blackout checkbox on the Global Overview dashboard, or REST API |
| *scheduled* | 開始時刻と期間が定義された時間制限のあるブラックアウト | Schedule Alert Blackout dialog, or REST API |
| *auto* | 到達不能なサーバーまたはエージェントのシステム開始のブラックアウト | Hourly system job (*pem.auto_blackout*) |

オブジェクトサーバーまたはエージェントは、アクティブなブラックアウトレコードが存在する場合、ブラックアウトしているとみなされます。さまざまなソースの複数のブラックアウトがオーバーラップする場合があります。オブジェクトは、最後のアクティブなレコードが終了するまでブラックアウトのままです。

#### 手動ブラックアウト

手動ブラックアウトは即時かつ無期限です。 Global Overview ダッシュボードの **Alert Blackout** チェックボックスを使用して、それらを作成およびクリアします。

#### 手動ブラックアウトの有効化

1. Global Overview ダッシュボードを開きます。
2. サーバーまたはエージェントテーブルで **Alert Blackout** 列を見つけます。
3. ブラックアウトするサーバーまたはエージェントのチェックボックスを選択します。

PEM はすぐに手動ブラックアウトレコードを作成します。 `alert_blackout` フラグは数秒以内に同期し、そのオブジェクトのアラート通知は停止します。チェックボックスを複数回選択しても効果はありません。PEM は、オブジェクトごとに 1 つの手動ブラックアウトレコードのみを作成します。

デフォルトでは、手動ブラックアウトは無期限です。 `max_manual_blackout` が構成されている場合、システムは制限を超過する手動ブラックアウトを自動的に期限切れにします。詳細は、**ブラックアウト動作の構成** を参照してください。

### 手動ブラックアウトの無効化

サーバーまたはエージェントの **Alert Blackout** チェックボックスをオフにします。

これにより、そのオブジェクトで現在アクティブな **すべての**ブラックアウト手動、スケジュール、および自動が終了します。アラート通知はすぐに再開されます。まだ始まっていない将来にスケジュールされた停電は影響を受けません。スケジュールされた開始時間にアクティブになります。

PEM アーカイブは、監査のために保存された元のスケジュールされた終了時刻を使用して、`pem.blackout_history` へのブラックアウトを終了しました。

### 計画停電

計画的な停電は、事前にメンテナンスウィンドウを定義します。PEM は、ウィンドウ中にのみアラートを抑制し、終了時に自動的に再開します。それらをクリアするために手動アクションは必要ありません。

### スケジュールされたブラックアウトの作成

1. **[ Management ] > [ Schedule Alert Blackout ]** を選択します。
2. ダイアログで、ブラックアウトするオブジェクトに応じて、**Servers** または **Agents** タブを選択します。
3. 右上隅にあるプラス記号 **+** を選択して、新しい行を追加します。
4. 次のフィールドに入力します。

- **Start time** — ブラックアウトが開始する日時。
- **Duration** — ブラックアウトの時間 1〜24 時間。
- **サーバー** または **エージェント** —含める 1 つ以上のオブジェクトを選択します。選択したすべてのオブジェクトは、同じブラックアウトウィンドウを共有します。

1. **Save** を選択します。

PEM は、選択したオブジェクトごとに 1 つのブラックアウトレコードを作成します。同じ保存操作のすべてのレコードは、共通のバッチ ID を共有し、ダイアログに単一の行として表示されます。

開始時刻が将来の場合、アラートはその時刻まで通常通り継続されます。開始時刻が現在または過去の場合、PEM はすぐにアラートを抑制します。

> **注釈**
>
> 保存されたブラックアウトレコードは不変です。ウィンドウの開始時間または期間を変更するには、レコードを削除し、新しいレコードを作成します。

### 計画的な停電の表示

**Management > Schedule Alert Blackout** を開くと、アクティブおよび保留中のスケジュールされた停電がすべて表示されます。`scheduled` ブラックアウトのみがこのダイアログに表示されます。手動および自動のブラックアウトは個別に管理されます。

### スケジュールされたブラックアウトの削除

1. **Management > Schedule Alert Blackout** を開きます。
2. 削除するブラックアウト行を選択します。
3. **Delete** 、次に **Save** を選択します。

これにより、バッチ内のすべてのレコードが削除されます。動作は、ブラックアウトが開始されたかどうかによって異なります。

- **まだ開始されていません** —PEM はブラックアウトをキャンセルします。履歴レコードは作成されません。
- **既にアクティブ** —ブラックアウトはすぐに終了し、PEM は `pem.blackout_history` にアーカイブします。

### 自動ブラックアウト

PEM は、到達不能になったサーバーとエージェントを自動的にブラックアウトし、インフラストラクチャがダウンした場合のアラートストームを防ぎます。

**Blackout unreachable servers/agents** システムジョブは 1 時間ごとに実行されます。各サーバーとエージェントの最後のハートビートタイムスタンプをチェックします。最後のハートビートが `server_contact_timeout` しきい値デフォルト 48 時間よりも古い場合、PEM は自動ブラックアウトレコードを作成します。タイムアウトウィンドウ内にオブジェクトが回復し、ハートビートが再開されると、PEM は自動ブラックアウトを自動的に終了します。

主な動作

- 自動ブラックアウトは、既にアクティブなブラックアウト手動、スケジュール、または自動があるオブジェクトをスキップします。重複するレコードは作成しません。
- サーバーが復旧すると、オートブラックアウトのみが終了します。アクティブな手動またはスケジュールされたブラックアウトは有効なままです。
- 自動ブラックアウトは、[ Schedule Alert Blackout ] ダイアログには表示されません。これらは `pem.blackout` または REST API `GET /api/v17/blackout/?source=auto` を介して可視されます。
- 自動ブラックアウトを完全に無効にするには、 `server_contact_timeout` を `0` に設定します。 **ブラックアウト動作の構成** を参照してください。

### 重複する停電

同じオブジェクトでさまざまなタイプのブラックアウトを同時にアクティブにすることができます。スケジュールされたブラックアウトを作成するときにオーバーラップガードはありません。同じまたは異なるソースの複数のブラックアウトは、同じオブジェクトに共存できます。オブジェクトは、最後のアクティブなレコードが終了するまでブラックアウトのままです。

- **例 — 手動+スケジュールされた共存**

スケジュールされたメンテナンスウィンドウが既にアクティブなときに、ユーザーは手動ブラックアウトを有効にします。両方のレコードが共存します。オブジェクトは、最後のものが終了するまでブラックアウトのままです。

- **例 — 手動オーバーライドを使用した自動ブラックアウト**

サーバーが到達不能になり、PEM は自動ブラックアウトを作成します。次に、管理者はブラックアウトも手動で有効にします。サーバーが復旧すると、自動ブラックアウトのみがクリアされます。手動は、管理者がチェックボックスをオフにするまで持続します。

- **例 — 将来のスケジュールされたウィンドウでチェックを解除する**

サーバーには、アクティブな手動ブラックアウトと、将来的にスケジュールされたメンテナンスウィンドウがまだ開始されていません。管理者はチェックボックスをオフにします。アクティブな手動ブラックアウトのみが終了します。将来のスケジュールは残り、その開始時刻にアクティブ化されます。

### ブラックアウト動作の構成

`pem.config` テーブルの次のパラメーターは、アラートのブラックアウト動作を制御します。

#### max_manual_blackout

| Setting | Value |
|---|---|
| Default | *NULL* (no limit) |
| Unit | hours |

手動ブラックアウトの最大期間を制御します。**システムジョブのプロセスアラートブラックアウト**によって実行時に適用されます。この設定を変更すると、既存のすべての手動ブラックアウトに遡って影響します。

| Value | Behavior |
|---|---|
| *NULL* or *0* | 制限なし — 手動ブラックアウトは無期限です |
| *> 0* | 手動ブラックアウトはこの数時間後に有効期限が切れます |

24 時間のキャップを設定するには

```
UPDATE pem.config SET value = 24 WHERE param = max_manual_blackout;
```

キャップを取り外すには

```
UPDATE pem.config SET value = NULL WHERE param = max_manual_blackout;
```

#### server_contact_timeout

| Setting | Value |
|---|---|
| Default | *48* |
| Unit | hours |

自動ブラックアウト機能を制御します。最後のハートビートがこのしきい値より古いサーバーとエージェントは、自動的にブラックアウトされます。

タイムアウトを 24 時間に変更するには

```
UPDATE pem.config SET value = 24 WHERE param = server_contact_timeout;
```

自動ブラックアウトを完全に無効にするには

```
UPDATE pem.config SET value = 0 WHERE param = server_contact_timeout;
```

#### blackout_history_retention

| Setting | Value |
|---|---|
| Default | *30* |
| Unit | days |

完了したブラックアウトレコードが pem.blackout_history に保持される期間を制御します。 **プロセスアラートブラックアウト**システムジョブは、この制限より古いレコードを自動的にパージします。

| Value | Behavior |
|---|---|
| *NULL* or *0* | 永久に保持します — 自動パージなし |
| *> 0* | この日数より古い履歴レコードをパージ |

履歴を 90 日間保持するには

```
UPDATE pem.config SET value = 90 WHERE param = blackout_history_retention;
```

### ブラックアウト履歴

PEM は、完了したブラックアウトを pem.blackout_history テーブルに自動的にアーカイブします。アーカイブされた各レコードには次のものが含まれます。

- actual_end_time — 停電が実際に終了したとき。
- scheduled_end_time —当初計画された終了時間早期に終了した停電の場合、 actual_end_time とは異なる場合があります。

完了したブラックアウトを直接照会するには

```
-- All completed blackouts
SELECT * FROM pem.blackout_history ORDER BY actual_end_time DESC;

-- Completed blackouts for a specific server (object_id = 5)
SELECT id, start_time, actual_end_time, source
FROM pem.blackout_history
WHERE object_type = 200 AND object_id = 5
ORDER BY actual_end_time DESC;
```

?include_history=true をリストエンドポイントに追加することにより、REST API を介して履歴を取得することもできます。アーカイブされたレコードはアクティブなレコードとともに返され、それらを区別するための archived: true フィールドが含まれます。

#### よくある質問

- **Alert Blackout チェックボックスのチェックを外しましたが、アラートは再開されていません。**

alert_blackout フラグは、チェックボックスをオフにするとすぐにクリアされます。数秒後もアラートが再開されない場合は、**プロセスアラートブラックアウト**システムジョブが有効になっていること、および PEM エージェントが実行されていることを確認します。

- **チェックボックスをオフにすると、スケジュールされた停電はキャンセルされますか？**

そのオブジェクトのすべての **現在アクティブな**ブラックアウト手動、スケジュール、および自動を終了します。まだ始まっていない将来にスケジュールされた停電は影響を受けません。スケジュールされた開始時間にアクティブになります。将来のスケジュールされたブラックアウトをキャンセルするには、Schedule Alert Blackout ダイアログまたは REST API を使用します。

- **特定のアラートのみをブラックアウトできますか?**

いいえ。ブラックアウトは、選択したサーバーまたはエージェント上のすべてのアラートに適用されます。アラートごとのブラックアウトの粒度はありません。

- **停電中に監視データはどうなりますか?**

プローブは通常通りデータの収集を継続します。アラートの評価と通知のみが抑制されます。ブラックアウト中に収集された履歴データは、ブラックアウトが終了すると確認できます。

- **定期的なブラックアウトたとえば、毎週日曜日の午前 2〜6 時などを作成できますか?**

UI から直接ではありません。各ブラックアウトは 1 回限りのウィンドウです。定期的なメンテナンスウィンドウの場合、cron ジョブまたは外部スケジューラーと REST API を使用して、ブラックアウトの作成を自動化します。

- **現在ブラックアウトしているオブジェクトを確認するにはどうすればよいですか?**

UI から、グローバル概要ダッシュボードの Alert Blackout チェックボックスは、各サーバーとエージェントの現在の状態を反映しています。SQL 経由

```
SELECT object_type, object_id, source, start_time, end_time
FROM pem.blackout
WHERE start_time <= now()
AND (end_time IS NULL OR end_time > now())
ORDER BY start_time;
```

### 8.3.3 Managing monitoring settings using profiles

プロファイルは、複数のサーバーとエージェントにわたって監視構成とアラートを管理するための集中化された再利用可能な方法です。

プロファイルは、特定のプローブ構成とアラート定義をバンドルする単一のテンプレートとして機能します。これらは、特定のサーバーまたはエージェントの個々の Manage Probes および Manage Alerts メニューを介して現在使用可能な構成と同じです。

ユーザーには少なくとも次のものが必要です。

- プロファイルを作成、更新、および削除する pem_manage_profile ロール。
- サーバーでプロファイルを割り当てまたは割り当て解除する pem_database_server_registration ロール。
- サーバーとエージェントの両方でプロファイルを割り当てまたは割り当て解除する pem_assign_profile ロール。

プロファイルの仕組み

- プローブ構成とアラート定義の特定のセットを使用してプロファイルを作成します。
- プロファイルを 1 つ以上のサーバーまたはエージェントに割り当てます。
- 割り当てられると、サーバーまたはエージェントは、プロファイルで定義された監視設定をすぐに継承します。
- プロファイルに変更が行われると、割り当てられたすべてのサーバーとエージェントに自動的に伝播されます。

プロファイルが割り当てられていないサーバーまたはエージェントは、個々のプローブ構成とアラート定義を使用し続けます。

#### 8.3.3.1 プロファイルを使用する目的

プロファイルを使用する主な目的は、PEM での多数のデータベースインスタンスの監視構成の管理に関連する管理負担を大幅に削減することです。

PEM は、単一のプラットフォームから多くのインスタンスを管理および監視するように設計されています。多数のサーバーを扱う場合、構成プローブとアラートがすべてのインスタンスで正しくて一貫していることを手動で確認することは、時間がかかり、エラーが発生しやすいプロセスになる場合があります。

この機能は、設定を再利用可能なプロファイルに一元化することにより、次のような価値を提供します。

- 時間の解放 - 複数のインスタンスにわたる構成管理を自動化し、管理者は監視ツールを管理するのではなく、データベース操作の最適化に集中できます。
- 一貫性の確保 - 大規模なサーバーフリートが目的の監視標準に準拠していることを保証します。
- エラーの削減 - 構成ミスを引き起こし、環境を効果的に監視する PEM の機能を損なう可能性のある人的エラーの可能性を最小限に抑えます。

#### 8.3.3.2 主要な概念

監視のさまざまな側面を管理するように設計された 2 種類のプロファイルがあります。

- **エージェントプロファイル** - PEM エージェントのプローブ構成とアラート定義を管理します。
- **サーバープロファイル** - 監視対象データベースサーバーのプローブ構成とアラート定義を管理します。

各プロファイルタイプの動作は次のとおりです。

| Status | Configuration source | Behavior |
|---|---|---|
| No Profile assigned | Default settings | サーバーまたはエージェントは、個々のプローブ構成とアラート定義を使用します。 |
| Profile assigned | Profile settings | サーバーまたはエージェントは、割り当てられたプロファイルで定義されたプローブおよびアラート設定を継承します。 |

- サーバーまたはエージェントにプロファイルを割り当てた後、エージェントまたはサーバーに既に割り当てられているプロファイルを変更しても、変更はすぐに適用されません。これらはドラフトバージョンとして保存されます。変更は、ドラフトの変更を適用することを明示的に選択した場合、割り当てられたサーバーまたはエージェントでのみ有効になります。
- `Revert` ボタンを使用して、ドラフトの変更を適用しない場合に破棄するオプションもあります。
- プロファイル管理サーバーまたはエージェントの個々の `Manage Probes` および `Manage Alerts` ページには、割り当てられたプロファイルから継承した現在の構成が表示されますが、これらのページで直接変更を行うことはできません。
- `Copy Probes` および `Copy Alerts` オプションは、非プロファイルサーバーまたはエージェントからプロファイル管理対象サーバーまたはエージェントに構成をコピーしようとする場合、使用できません。

#### 8.3.3.3 管理プロファイルへのアクセス

`Manage Profiles` セクションは、PEM 環境全体で再利用可能なモニタリング設定を作成、構成、および展開するための集中インターフェイスを提供します。

- PEM コンソールの `Configuration` メニューの下の `Manage Profiles` にアクセスできます。 `Manage Profiles` タブは、オブジェクトツリーで選択したオブジェクトとは無関係に動作します。From here, you can:
- **プロファイルの作成**
- **プロファイルのクローンを作成する**
- **プロファイルの編集**
- **プロファイルの割り当て**
- **プロファイルの割り当て解除**
- **プロファイルの削除**
- **プロファイルのエクスポート**
- **プロファイルのインポート**

メインテーブルには、各プロファイルに関する重要な情報が表示されます。

| Column name | Description. |
|---|---|
| Name | The unique identifier assigned to the profile. |
| Type | The category of the profile, such as Agent profile or Server profile. |
| Active Assignments | The count of servers or agents currently assigned to the profile. |
| Apply/Revert | Displays the synchronization status of the profile: - Apply (Green button) - Deploys the draft changes made to the profile to all currently assigned servers and agents. - Revert (Red loop button) - Discards any non-applied draft changes, resetting the profile to the last successfully deployed version. - Green Tick Icon - Indicates that there are no pending draft changes; all configurations are synced with the assigned objects. |

### プロファイルのエクスポート

`pem_manage_profile` ロールを持つユーザーは、1 つ以上のプロファイルをポータブルな `.pemprf` JSON バンドルとしてエクスポートできます。

[ **Manage Profiles** ] タブで

1. 行チェックボックスを使用して 1 つ以上のプロファイルを選択します。
2. 少なくとも 1 つの行が選択されると、 **Export** ボタンがアクティブになります。
3. **Export** を選択します。
4. `Export_Profiles_<timestamp>.pemprf` という名前のファイルがダウンロードされます。

`.pemprf` ファイルは、トップレベルの `version` フィールドと、エクスポートされたプロファイル構成を含む `profiles` 配列を含む JSON ドキュメントです。

選択したプロファイルに未保存のドラフト編集がある場合、エクスポートされたファイルには、ドラフトではなくプロファイルの**公開** 状態が含まれます。

すべてのプロファイル設定は、エクスポートされたバンドルに含まれません。

- 次の設定は、エクスポートされたバンドルに保存されます。
- 名前、説明、ターゲット kind
- 周波数、保持、ステータスなどのプローブ構成有効/無効
- アラートしきい値、演算子、およびパラメーター `auto_created` flag
- スクリプト実行設定 ( `execute_script` 、`execute_script_on_clear` 、`execute_script_on_pem_server` ) とスクリプトコード本体
- 次の設定は削除され、インポート後に再構成する必要があります。
- 通知ルーティング設定 `email_group_id` 、重大度ごとのメールグループ、Webhook ID、SNMP トラップバージョン、および Nagios サービス
- 通知有効フラグ `send_email` 、重大度ごと `send_trap` 、`submit_to_nagios` 、`send_notification` 、`override_default_config` 、および `cleared_alert_enable`

プロファイルのインポート後、アラートが評価されてダッシュボードに表示されますが、**各チャネルを再度有効にして、アラートダイアログで宛先を構成し、**適用*をクリックするまで、メール、SNMP、Webhook、または Nagios 通知は配信されません** *。スクリプト実行設定はラウンドトリップ後も存続し、再入力する必要はありません。

> **警告**
>
> *.pemprf* ファイルには、資格情報、トークン、または環境固有のパスを含むスクリプトコードが含まれている場合があります。それらを機密のアーティファクトとして扱います。レビューなしで電子メールで送信したり、パブリックまたは共有リポジトリにコミットしたりしないでください。

### プロファイルのインポート

`pem_manage_profile` ロールを持つユーザーは、`.pemprf` バンドルからプロファイルをインポートできます。

[ **Manage Profiles** ] タブで

1. ツールバーで **Import** を選択します。
2. アップロードする `.pemprf` ファイルを選択します。
3. **Skip installation?** チェックボックスを有効にするかどうかを選択して、名前の競合の処理方法を制御します。
4. プロファイルごとの結果をレポートする **Import Summary** テーブルを確認します。

**Skip stable?** チェックボックスは、ターゲットインスタンスに名前が既に存在するプロファイルをインポートが処理する方法を制御します。

| Scenario | Result |
|---|---|
| Profile name is new | Imported as a published profile. |
| Profile name exists and **Skip existing?** is checked | Skipped. The summary reports "profile name already exists." |
| Profile name exists and **Skip existing?** is unchecked | Imported as a **draft** on the existing profile. Select **Apply** on that row to publish the draft and apply the changes to assigned targets. |

インポートは、型の不一致と欠落している参照を次のように処理します。

- インポートされたプロファイルと既存のプロファイルたとえば、サーバーとエージェントで `target_kind` が異なる場合、そのプロファイルのインポートは説明的なエラーで失敗します。
- バンドルで参照されるプローブまたはアラートテンプレートがターゲット PEM インスタンスに存在しない場合、その項目はスキップされます。プロファイルの残りの部分は正常にインポートされます。概要テーブルには、スキップされた項目がリストされています。
- 参照があいまいな場合ターゲットインスタンスで 2 つのアクティブな一致)、インポートはエラーで拒否されます。

### プロファイルの作成

pem_manage_profile ロールを持つユーザーまたはスーパーユーザーがプロファイルを作成できます。

Manage Profiles タブで、+ を使用して新しいプロファイルを追加します。

1. [新規追加] ボタンをクリックします。
2. ダイアログボックスがポップアップして、ターゲットタイプサーバーまたはエージェントを選択します。ターゲットタイプを選択すると、そのプロファイルで変更することはできません。Next をクリックします。
3. 新しいダイアログが開き、構成オプションの 3 つのタブが開きます。

- **General** - プロファイルの一意の名前とオプションの説明を入力します。
- **Probes** - プロファイルに含めるプローブを選択および構成します。サーバーレベルで選択できるのは、サーバー、データベース、スキーマ、エージェント、および拡張レベルのプローブのみです。
- **アラート** - プロファイルに含めるアラートを選択および構成します。サーバーおよびエージェントレベルのアラートのみが選択できます。

4. プロファイルを構成したら、Save をクリックして作成します。新しいプロファイルが Manage Profiles テーブルに表示されます。

### プロファイルのクローンを作成する

pem_manage_profile ロールまたはスーパーユーザーを持つユーザーは、既存のプロファイルを複製して、新しいプロファイルの開始点として使用できます。

[ **Manage Profiles** ] タブで

1. **Add New** ボタンをクリックします。
2. ダイアログボックスが表示され、ターゲットタイプサーバーまたはエージェントを選択します。**Next** をクリックします。
3. ダイアログの **General** タブで、選択したタイプの既存のプロファイルをリストする **Copy from** ドロップダウンを使用して、ソースプロファイルを選択します。

ソースプロファイルを選択すると、新しいプロファイルの **Name** Copy of <source> として推奨、**Description** 、 **Probes** 、および **Alerts** タブが事前に設定されます。すべての値は、保存する前に編集可能なままです。

1. 必要に応じて、 **Probes** および **Alerts** タブで事前入力された設定を確認および調整します。
2. **Save** をクリックして、クローンを作成します。新しいプロファイルが **Manage Profiles** テーブルに表示されます。

クローンプロファイルはソースから完全に独立しています。

- ソースへの親/子リンクのない公開プロファイルとして作成されます。
- ソースからサーバーまたはエージェントの割り当てを継承しません。
- ソースを編集または削除しても、クローンには影響しません。その逆も可能です。

- ソースからコピーされた自動作成アラートは、通常に作成されるプロファイルと同様に、クローンでは削除できないままです。

プロファイルタイプサーバー ↔エージェントを切り替えると、 **コピー元**選択がクリアされます。サーバープロファイルは、エージェントプロファイルのソースとして使用できません。**Copy from** 選択をクリアすると、 **Probes** および **Alerts** タブがデフォルトにリセットされます。

### プロファイルの編集

Manage Profiles タブで

- 変更するプロファイルの横にある Edit icon をクリックします。
- 編集中にプロファイル名を更新することはできません。
- プロファイルの説明、プローブ構成、およびアラートは変更できます。
- プロファイルにアクティブな割り当てがある場合、 Apply および Revert ボタンを使用して、ドラフトの変更を展開または破棄できるようになります。

Apply ボタンをクリックしたときにのみ、変更は割り当てられたすべてのサーバーおよびエージェントと同期されます。Revert をクリックすると、プロファイルに加えられたすべての変更が以前のステータスにリセットされます。

### プロファイルの割り当て

pem_assign_profile または pem_database_server_registration ロールを持つユーザーは、サーバーまたはエージェントにプロファイルを割り当てることができます。

PEM コンソールのオブジェクトツリーで

- サーバーまたはエージェントを右クリックします。
- コンテキストメニューから Properties を選択します。
- [ Properties ダイアログで、 General タブに移動します。
- Assigned Profile フィールドを見つけます。
- ドロップダウンメニューを使用して、サーバーまたはエージェントに割り当てる目的のプロファイルを選択します。
- Save をクリックして変更を適用します。

### プロファイルの割り当て解除

pem_assign_profile ロールを持つユーザーはサーバーまたはエージェントからプロファイルの割り当てを解除でき、pem_database_server_registration ロールを持つユーザーはサーバーからプロファイルの割り当てを解除できます。

オブジェクトツリーで

- サーバーまたはエージェントを右クリックします。
- コンテキストメニューから Properties を選択します。

- [ Properties ダイアログで、 General タブに移動します。
- Assigned Profile フィールドを見つけます。
- ドロップダウンメニューを使用して<None> を選択し、サーバーまたはエージェントからプロファイルの割り当てを解除します。
- Save をクリックして変更を適用します。

サーバーまたはエージェントからプロファイルの割り当てを解除する場合

- プローブはデフォルトの構成にリセットされます。
- アラートは残りますが、プロファイルで管理されなくなります。
- 各タブからサーバーまたはエージェントのプローブとアラートを個別に管理できるようになりました。

#### プロファイルの削除

[ **Manage Profiles** ] タブで

1. 削除するプロファイルの横にある **Delete アイコン** を選択します。
2. プロファイルにアクティブな割り当てがない場合、確認ダイアログが表示され、削除を完了します。
3. プロファイルがサーバーまたはエージェントにアクティブに割り当てられている場合、**Reassign Profile** ダイアログが表示され、監視構成の損失を防ぎます。
4. ダイアログで、ドロップダウンメニューを使用して置換プロファイルを選択するか、ターゲットをシステムのデフォルト構成にリセットするオプションを選択します。
5. **Save** を選択して再割り当てを適用し、元のプロファイルを削除します。

再割り当てプロセスは、 **サーバー** と **エージェント**の両方のターゲットタイプに適用されます。

#### 削除と再割り当ての権限

プロファイルを削除および再割り当てできる機能は、割り当てられた PEM ロールによって異なります。

- pem_admin ロールを持つユーザーは、プロファイルの削除、既存のターゲットへの再割り当て、またはデフォルトにリセットできます。
- pem_manage_profile ロールを持つユーザーは、未割り当てのプロファイルを削除するか、割り当てられたプロファイルをデフォルトにリセットすることにより削除できます。新しいプロファイルを既存のターゲットに再割り当てすることはできません。
- pem_manage_profile および pem_database_server_registration の両方のロールを保持するユーザーは、プロファイルを削除し、新しいターゲットに再割り当てする完全な権限を持っています。

### 8.3.3.4 プロファイルの割り当てページ

**Profile Assignments** ページでは、PEM 環境内のすべての監視対象サーバーとエージェントを単一のビューで提供します。これを使用して、各サーバーまたはエージェントに個別に移動せずに、モニタリングプロファイルを一括、グループごと、または一度に 1 つずつ割り当てまたは割り当て解除します。

ページを開くには、 **Management > Monitoring > Profile Assignments** に移動します。

このページにアクセスするには、pem_assign_profile ロールが必要です。 pem_assign_profile を継承するロール—pem_manage_profile 、pem_database_server_registration 、および PEM 管理ロールを含む—にもアクセスします。これらのロールがないと、メニューに**プロファイルの割り当て**オプションが表示されず、ダイレクトナビゲーションに**アクセスが拒否されました** と表示されます。

### このページに示されている内容

このページには、登録されているすべてのサーバーとエージェントのツリーがサーバーグループごとに表示されます。クラスターの一部であるサーバーは、クラスターノード内にネストされて表示されます。各行には、サーバーまたはエージェントの名前と、現在割り当てられているプロファイルが表示されます。

上部の統計カードは、X / Y profiled として表現されるプロファイルが割り当てられているサーバーとエージェントの数を示します。

### プロファイルの割り当てと割り当て解除

#### インライン代入

単一のサーバーまたはエージェントのプロファイルを変更するには、その行のプロファイルドロップダウンを使用し、必要なプロファイルを選択します。

#### グループの割り当て

サーバーグループのすべてのメンバーにプロファイルを一度に割り当てるには、グループのドロップダウンを使用します。サーバーとエージェントの両方を含むグループの場合、ドロップダウンは個別の**サーバープロファイルの設定** および **エージェントプロファイルの設定**アクションを提供します。1 つを選択すると、グループ内のそのタイプの現在のメンバーすべてにプロファイルが適用されます。

> **注釈**
>
> グループの割り当ては、グループの現在のメンバーに 1 回限りの一括適用です。永続的な継承ルールは作成しません。後でグループに追加されたサーバーまたはエージェントは、プロファイルを自動的に受信しません。サーバーまたはエージェント**のプロパティ**のオブジェクトごとの**割り当てプロファイル**ドロップダウンは、常に最後に行われた割り当てを反映します最後の書き込みが優先します。

#### 大量の割り当て

プロファイルを一度に複数のターゲットに割り当てるには、チェックボックスを使用して行を選択し、**Assign to N** を選択します。ダイアログは選択に基づいて調整されます。

- サーバーのみの選択 サーバープロファイルを聞いてきます。
- エージェントのみの選択 エージェントプロファイルを聞いてきます。
- 混合選択サーバープロファイルとエージェントプロファイルの両方を要求します。

多数のターゲットを選択すると、確認ノートが表示されます。

#### プロファイルの割り当ての解除

サーバーまたはエージェントからプロファイルを削除するには、プロファイルドロップダウンで **Unassigned** を選択します。大量の割り当てを解除するには、行を選択して **Assign to N** を使用し、ダイアログで **Unassigned**

を選択します。

#### 検索とフィルタリング

検索バーを使用して、名前でサーバーまたはエージェントを見つけます。フィルタドロップダウンを使用して、特定のプロファイルが割り当てられたターゲットにツリーを絞り込むか、未割り当てのターゲットのみを表示します。検索とフィルターを組み合わせることができます。

#### 監査ログ

PEM は、 `pem.server` および `pem.agent` のデータベーストリガーを介して `pem.event_history` にプロファイルの割り当ての変更を自動的に記録します。インライン割り当てと大量割り当ての両方が監査レコードを生成します。コンプライアンスまたはトラブルシューティングに PEM のイベント履歴を使用する場合、そこを参照して、プロファイルの割り当てを誰がいつ変更したかを追跡します。

### 8.3.4 Notifications

アラートがトリガーまたはクリアされた場合、PEM は通知を送信するか、スクリプトを実行できます。次のオプションを使用して通知を送信できます。

- *SMTP*
- *Webhooks*
- *SNMP++*
- *Nagios*

#### 8.3.4.1 SMTP

**重要**

PEM がメール通知を送信する前に、SMTP サーバーを使用して電子メールを配信するように PEM サーバーを構成する必要があります。

**注釈**

PEM は、お客様が提供および保守する SMTP サーバーを介して電子メールを送信します。メールサーバーのセットアップと管理は、PEM インストールの範囲外です。

#### SMTP 設定の構成

PEM が電子メール通知を送信する前に、関連するサーバー構成パラメーターを設定して SMTP サーバーに接続する必要があります。

SMTP 設定を構成するには

1. PEM ブラウザーで、 **Management > Server Configuration** を選択します。

2. SMTP サーバーと一致するように次のパラメーターを設定します。

| Parameter | Default | Description |
|---|---|---|
| *smtp_server* | *127.0.0.1* | SMTP サーバーのホストアドレス。 |
| *smtp_port* | *25* | SMTP サーバーがリッスンするポート。 |
| *smtp_enabled* | *true* | メール送信を有効にするには、`true`に設定します。 |
| *smtp_encryption* | *false* | 暗号化された接続を介してメールを送信するには、`true`に設定します。 |
| *smtp_authentication* | *false* | SMTP サーバーが認証を必要とする場合、`true`に設定します。 |
| *smtp_username* | — | SMTP サーバーで認証するユーザー名。`smtp_authentication`が`true`の場合に必須。 |
| *smtp_password* | — | SMTP サーバーで認証するためのパスワード。`smtp_authentication`が`true`の場合に必須。 |

3. **Save** を選択して変更を適用します。

SMTP 関連パラメーターの完全なリストについては、*Server configuration parameters — reference* を参照してください。

### メールグループの作成

PEM は、注意が必要な条件がないかシステムを監視します。電子メールグループを使用して、サーバーが現在の値がアラート定義で指定されたしきい値から逸脱した場合に通知するユーザーの電子メールアドレスを指定できます。電子メールグループは、指定した期間に複数のユーザーに通知するか、特定のユーザーをターゲットにできます。

**Send the notifications** タブを使用して、SMTP メール受信者のグループを構成します。 [ **Send the notifications** ] タブを開くには、PEM クライアントで、[ **Management > Alerts > Manage Alerts** ] を選択します。 [ **Manage Alerts** ] タブが開いたら、[ Quick Links ] ツールバーから [ **Email Groups** ] を選択します。

[ **Send the notifications** ] タブには、現在定義されているメールグループのリストが表示されます。既存のグループを変更するには、グループ名を選択し、行の左端にある **Edit** を選択します。

新しいメールグループを定義するには、 **メールグループ**テーブルの右上隅にあるプラス記号+を選択します。 [ Email Groups ] ダイアログボックスを使用して、電子メールグループとそのメンバーを定義します。

電子メールグループ定義の各行は、一連の電子メールアドレスを特定の期間に関連付けます。アラートがトリガーされると、サーバーは各行で指定された時間を評価し、アラートがトリガーされた時間に定義が関連付けられているグループメンバーにメッセージを送信します。

1. **Group Name** フィールドにメールグループの名前を入力します。
2. [ **Options** ] タブを開くには、グループメンバーテーブルでプラス記号+を選択します。
3. 指定した期間に通知を受信するようにメンバーアドレスを追加します。
    - **Reply to addresss** フィールドに受信者アドレスのコンマ区切りリストを入力します。

- コピーを受信するアドレスのコンマ区切りリストを入力します。 **CC Addresses** フィールドのメール。
- **Bcc Addresses** フィールドに他の受信者に知られることなくメールのコピーを受信するアドレスのコンマ区切りリストを入力します。
- **From Address** フィールドからメッセージを送信します。
- **Subject prefix** フィールドを使用して、通知が送信されるときに各件名行の先頭に追加するメッセージを提供します。
- **From Time を使用する** および **To Time** セレクターを使用して、この行で指定されたグループメンバーへの通知の時間範囲を指定します。これらの値を PEM クライアントホストのロケールで提供します。PEM サーバーは、必要に応じて時間を他のタイムゾーンに変換します。

4. **Add** を選択してテーブルに行を追加し、別の期間とそれらの時間帯に通知する電子メールアドレスを指定します。
5. メールグループの定義が終了したら、[ **Save** ] を選択します。

メールグループを作成した後、 **Manage Alerts** タブを使用して、アラートの通知の詳細を設定し、グループに通知を転送できます。

### メールグループの削除

電子メールグループを削除するには、電子メールグループテーブルでグループの名前を選択し、グループ名の左側にある **Delete** を選択します。

グループ名は [電子メールグループ] テーブルに赤で表示されます。 **Save** を選択して、テーブルからグループを完全に削除します。

### メールテンプレートのカスタマイズ

PEM は、ユーザーの注意を必要とする条件がシステムを監視し、電子メール通知を送信します。**電子メールテンプレート**を使用して、電子メールの件名と、現在の値がアラートで指定されたしきい値から逸脱している場合にサーバーが送信するペイロードをカスタマイズします。

これらは、カスタマイズ可能なデフォルトのメールテンプレートです。

- アラートが検出されました
- 警戒レベルが上昇しました
- 警戒レベルの低下
- アラートがクリアされました
- アラートリマインダー
- 仕事の成功
- ジョブの失敗
- ジョブのキャンセル

・ジョブステップ

・ジョブステップデータベースサーバ

電子メールテンプレートをカスタマイズするには、電子メールテンプレート名の横にある edit ボタンを選択します。必要に応じて、使用可能なプレースホルダーのリストから **Subject** および **Payload** フィールドを編集します。フィールド値を変更した後、[ **Save** ] を選択します。

デフォルトのメールテンプレートがカスタマイズされている場合、**メールテンプレート** タブの `Custom Template?` 列にそのテンプレートの緑色のチェックマークが表示されます。

カスタマイズされたメールテンプレートをデフォルトのテンプレートとして復元するには、カスタムテンプレートを選択し、**メールテンプレート** タブの上部にある元に戻すボタンを選択します。

#### 8.3.4.2 Webhooks

Webhook を使用して、構成済みのアプリケーションのしきい値違反に関するアラートイベントの通知を受信するように PEM サーバーを構成する必要があります。

##### Webhook の作成

PEM は、ユーザーの注意を必要とする条件がシステムを監視します。Webhook を使用して、現在の値がアラート定義で指定されたしきい値から逸脱した場合に通知を受信するエンドポイントを作成できます。トリガーされたイベントに基づいて、PEM は複数の Webhook エンドポイントまたは特定のターゲット Webhook エンドポイントに通知を送信します。

**Webhooks** タブを使用して、エンドポイントの受信者を構成します。 [ **Webhooks** ] タブを開くには、 **Management > Alerts > Manage Alerts** を選択します。 [ **Manage Alerts** ] タブの [Quick Links] ツールバーで、[ **Webhooks** ] を選択します。

[ **Webhooks** ] タブには、エンドポイントとして現在定義されている受信者アプリケーションのリストが表示されます。エンドポイントを選択し、行の左端にある **Edit** を選択して、既存のエンドポイントを変更します。

新しい Webhook を定義するには、テーブルの右上隅にあるプラス記号+を選択します。次に、**General** タブを使用して、Webhook の基本的な詳細を定義できます。

- Webhook の名前を **Name** フィールドに入力します。
- **URL** フィールドに、すべての通知を配信する Webhook URL を指定します。
- **Request Method** フィールドで、呼び出しに使用される要求方法タイプを設定します **POST** または **PUT** 。
- デフォルトでは、Webhook は有効になっています。Webhook を無効にするには、 **Enable?** を **No** に設定します。

> **注釈**
>
> **Enable?**設定は、*agent.cfg* ファイルで `enable_webhook` パラメーターが `true` に設定されている場合にのみ機能します。デフォルトでは、*enable_webhook* パラメーターは、PEM サーバーホストで実行

されているエージェントに対してのみ`true` に設定されます。他のホストで実行されている他のすべてのエージェントの場合、手動で `true` に設定する必要があります。

### Webhook SSL 構成の定義

Windows の各エージェント構成ファイルまたはレジストリで Webhook SSL パラメーターを定義できます。Webhook SSL パラメーターのリストは *Modifying agent configuration* にあります。エージェント構成パラメーターを追加または削除した場合、エージェントを再起動してそれらを適用する必要があります。

- Windows システムでは、Webhook の PEM レジストリエントリは `HKEY_LOCAL_MACHINE\Software\EnterpriseDB\PEM\agent\WEBHOOK` にあります。
- Linux システムでは、Webhook の PEM 構成オプションは `agent.cfg file` に保存されます。デフォルトで`/usr/edb/pem/agent/etc` にあります。

```
[WEBHOOK/Django]
webhook_ssl_key=<webhook_client_ssl_key_path>
webhook_ssl_crt=<webhook_client_ssl_certificate_path>
webhook_ssl_ca_crt=<webhook_server_ca_certificate_path>
webhook_ssl_crl=<crl_file_path_to_validate_webhook_server>
allow_insecure_webhooks=<true|false>
```

図 8: Example - Webhook SSL Parameters in agent.cfg file

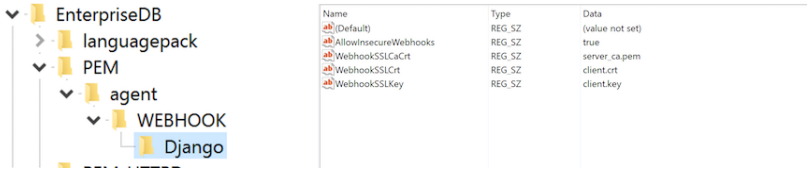



図 9: Example - Webhook SSL Parameters in windows registry

### Webhook ヘッダーの定義

[ **HTTP Headers** ] タブを使用して、Webhook エンドポイントの呼び出し中に渡すヘッダーパラメーターを定義します。

- すべての値をキーと値のペアとして指定します。
- **Key** フィールドにキーパラメーターを指定し、 **Value** フィールドに値を指定します。
- HTTP ヘッダーを追加するには、HTTP Headers テーブルの右上隅にあるプラス記号+を選択します。
- HTTP ヘッダーを削除するには、 **Key** の左側にある **Delete** を選択します。ヘッダーはリストに残りますが、取り消し線フォントになります。**Save** を選択して、ヘッダーを完全に削除します。
- HTTP ヘッダーを編集するには、 **Key** の左側にある **Edit** を選択します。

### Webhook ペイロードの定義

**Payload** タブを使用して、アラートがトリガーされたときにエンドポイントに送信する JSON データを定義します。

- **Type** は、形式タイプ JSON で送信するデータを指定します。

- **テンプレート**を使用して、エンドポイントに送信される JSON データを構成します。テンプレートでは、次のプレースホルダーを使用できます。
- `%AlertID%` - トリガーされたアラートの ID。
- `%AlertName%` - トリガーされたアラートの名前。
- `%ObjectName%` - アラートがトリガーされたサーバーまたはエージェントの名前。
- `%ObjectType%` - アラートが生成されたタイプ。
- `%ThresholdValue%` - アラートがトリガーされたときにメトリックが到達したしきい値。
- `%CurrentValue%` - アラートをトリガーしたメトリックの現在値。
- `%CurrentState%` - アラートの現在の状態。
- `%OldState%` - アラートの以前の状態アラートが生成されたエージェントの名前。
- `%ServerID%` - アラートが生成されたサーバーの ID。
- `%ServerName%` - アラートが生成されたサーバーの名前。
- `%ServerIP%` - アラートが生成されたサーバーの IP またはアドレス。
- `%ServerPort%` - アラートが生成されたサーバーのポート。
- `%DatabaseName%` - 名前アラートが生成されたデータベースの。
- `%SchemaName%` - アラートが生成されたスキーマの名前。
- `%PackageName%` - アラートが生成されたパッケージの名前。
- `%DatabaseObjectName%` - データベースオブジェクトの名前、アラートが生成されたテーブル名またはファンクション名など。
- `%Parameters%` - アラートを生成するために使用されるカスタムパラメーターのリスト。
- `%AlertInfo%` - アラートの詳細なデータベースオブジェクトレベル情報。
- **Test Connection** を選択して、記載されたエンドポイントへの通知配信をテストします。

### Webhook アラートレベルの定義

**Notifications** タブを使用して、Webhook エンドポイントのアラートレベルを指定します。

- すべてのアラートレベルで通知を送信できるようにするには、**すべてのアラート** を **Yes** に設定します。
- 特定のアラートレベルに到達したときに通知を送信するには、アラートレベルの横にあるスライダーを **Yes** に設定します。個々のアラートレベルを構成するには、**すべてのアラート** を **No** に設定する必要があります。

### 例 slack に通知を送信する

Slack では、Getting started with incoming webhooks の指示に従って、次のことを行います。

- Slack アプリを作成します。
- そのアプリの着信 Webhook をアクティブにします。

- 選択したチャンネルまたはユーザーに投稿する Webhook を追加します。

新しい Webhook には、`https://hooks.slack.com/services/x/y/z` のような一意の URL があります。この URL に通知を送信するように PEM を構成できるようになりました。

PEM *Webhook の作成* で、わかりやすい名前を付け、前に取得した URL を **URL** フィールドにコピーします。 **Request Method** が **POST** に設定され、**Enable?** が **Yes** に設定されていることを確認します。 **Alert Notifications** の下のすべてのスライダーを **Yes** に設定します。

HTTP ヘッダーの下に、キー `Content-Type` と値 `application/json` を使用してヘッダーを追加します。

**Payload** の下で、デフォルトのテンプレートを削除し、次の例のように `text` をトップレベルキーとしてテンプレートを指定します。

```
{"text": "%AlertName% on %ObjectType% %ObjectName% is now %CurrentState%"}
```

これで、接続をテストできます。成功した場合、PEM は通知を発行し、指定したテンプレートがメッセージとして Slack チャネルに表示されます。

Webhook を保存し、通常のように PEM の使用を続行します。PEM はすべてのアラートを Slack チャネルに送信するようになりました。

### Webhook の削除

削除の Webhook をマークするには、Webhooks テーブルで Webhook 名前を選択し、名前の左側にある **Delete** を選択します。アラートはリストに残りますが、取り消し線フォントになります。

**Delete** はトグルです。 2 度目に **Delete** を選択することにより、削除を元に戻すことができます。 **Save** を選択して、Webhook 定義を完全に削除します。

### OAuth を使用した Webhook 要求の認証

Webhook エンドポイントに OAuth が構成されている場合

1. 通知を送信する前に、PEM は OAuth プロバイダーからアクセストークンを要求します
2. トークンは自動的にキャッシュされ、有効期限が切れるまで再利用されます
3. アクセストークンは、要求に `Bearer` トークンとして含まれます
4. 認証が失敗した場合、PEM は自動的にトークンを更新し、再試行します

#### 前提条件

Webhook の OAuth を構成する前に、次のものが必要です。

1. PEM の既存の Webhook エンドポイント
2. アイデンティティプロバイダーからの OAuth クライアント資格情報

- **クライアント ID**
- **クライアントシークレット**
- **トークン URL** 承認サーバーエンドポイント

3. オプション **Scopes** および **Audience** 値プロバイダーによって異なります。
4. PEM ホストにアクセスして pemEncryptor ユーティリティを実行する
5. pem_admin ロールを持つユーザーまたはスーパーユーザーとして PEM バックエンドデータベースにアクセスします。

### 資格情報の暗号化

client_id および client_secret は、データベースに保存する前に暗号化する必要があります。PEM サーバーの/usr/edb/pem/bin/ にある pemEncryptor ユーティリティを使用します。

#### ステップ 1 クライアント ID を暗号化する

```
export PEM_ENC_PAYLOAD=your-client-id
/usr/edb/pem/bin/pemEncryptor
```

次のセクションで使用する暗号化された出力を保存します。

#### ステップ 2 クライアントシークレットを暗号化する

```
export PEM_ENC_PAYLOAD=your-client-secret
/usr/edb/pem/bin/pemEncryptor
```

次のセクションで使用する暗号化された出力を保存します。

### OAuth 構成の追加

SQL を使用して、既存の Webhook の OAuth を構成します。<encrypted_client_id> および<encrypted_client_secret> を pemEncryptor から取得した値に置き換えます。

```
INSERT INTO pem.webhook_oauth_config (
    webhook_id,
    token_url,
    client_id,
    client_secret,
    scope,
    audience,
    auth_method
)
VALUES (
    <webhook_id>,
    <token_url>,
    <encrypted_client_id>,
    <encrypted_client_secret>,
    <scopes>,                   -- Optional, can be NULL
    <audience>,                 -- Optional, can be NULL
    client_secret_basic         -- Optional, defaults to client_secret_basic
);
```

### 構成パラメーター

| Parameter | Required | Default | Description |
|---|---|---|---|
| *webhook_id* | Yes | — | Webhook エンドポイントの ID |
| *token_url* | Yes | — | OAuth トークンエンドポイント URL |
| *client_id* | Yes | — | OAuth クライアント ID 暗号化する必要があります |
| *client_secret* | Yes | — | OAuth クライアントシークレット 暗号化する必要があります |
| *scope* | No | NULL | スペースで区切られた OAuth スコープ |
| *audience* | No | NULL | ターゲット API 識別子設定されていない場合は自動派生 |
| *auth_method* | No | *client_secret_basic* | 資格情報がトークンエンドポイントに送信される方法以下を参照してください。 |

### 認証方法 RFC 6749

PEM は、RFC 6749 セクション 2.3 で定義された 2 つの標準のクライアント認証方法をサポートしています。

| Value | Description | When to use |
|---|---|---|
| *client_secret_basic* | HTTP Basic Auth ヘッダーを介して送信される資格情報 RFC 推奨 | Default — works with most providers |
| *client_secret_post* | POST 要求本文で送信される資格情報 | Use only when your provider requires it (e.g., some custom OAuth servers) |

**注釈**

ほとんどのプロバイダー Auth0、Okta、Azure AD、Keycloak は、両方の方法をサポートしています。AWS Cognito には `client_secret_basic` が必要です。よくわからない場合は、デフォルトを使用します。

### 構成の更新

```
- - Update scopes
UPDATE pem.webhook_oauth_config
SET scope = api:read api:write,
    modified_on = CURRENT_TIMESTAMP
WHERE webhook_id = 1;

- - Update token URL
```

(次のページに続く)

(前のページからの続き)

```
UPDATE pem.webhook_oauth_config
SET token_url = https://new-auth.example.com/oauth/token,
    modified_on = CURRENT_TIMESTAMP
WHERE webhook_id = 1;
```

#### OAuth 構成の削除

Webhook の OAuth 認証を無効にするには

```
DELETE FROM pem.webhook_oauth_config WHERE webhook_id = 1;
```

### 8.3.4.3 SNMP

通知を送信する前に、SNMP トラップ/通知レシーバーに通知を送信するように PEM サーバーを構成する必要があります。SNMP バージョンを識別するには、SNMP バージョンを v1、v2、または v3 に設定します。

#### 例-net-snmp トラップレシーバーを使用して SNMP V3 トラップを構成する

1. [サーバーの構成] ダイアログボックスでプレーンテキスト形式で snmp_security_engine_id ～PEM_SNMP_AGENT を設定します。
2. プレーンテキスト値を 16 進形式に変換して、snmptrapd.conf ファイルで使用します。snmp_security_engine_id の 16 進値は、最大 32 オクテット長まで指定できます。

```
echo PEM_SNMP_AGENT | hexdump -v -e /1 "%02X"
50454D5F534E4D505F4147454E540A
```

1. Server Configuration ダイアログボックスで次のパラメーターを設定します。
   - **snmp_security_nam** から **pem_snmp_user**
   - **snmp_authentication_protocol** から **MD5**
   - **snmp_authentication_password** から **pem_auth_pass**
   - **snmp_privacy_protocol** から **DES**
   - **snmp_privacy_password** から **pem_priv_pass**
2. snmptrapd.conf ファイルには次の値があります。

```
createUser -e 0x50454D5F534E4D505F4147454E540A pem_snmp_user MD5 pem_auth_pass DES␣
↪pem_priv_pass
authUser log pem_snmp_user
```

> **注釈**
>
> エージェントを root として実行して、SNMP V3 で通知を送信できるようにする必要があります。

#### 8.3.4.4 Nagios

PEM サーバーは、ユーザー定義のアラートがトリガーされると、パッシブアラート結果を Nagios ネットワークアラートソフトウェアに送信できます。トリガーされたアラートを Nagios に通知するように PEM サーバーに指示するには、次のことを行う必要があります。

- PEM サーバーから Nagios への通知をトリガーする各アラートの Nagios 通知を有効にします。`host.cfg` および `services.cfg` ファイルを作成する前に、PEM アラートを構成する必要があります。
- PEM サーバーの Nagios 関連の動作を構成します。
- `host.cfg` および `services.cfg` 構成ファイルを作成します。
- 必要に応じて、Nagios 構成ファイルを変更し、Nagios サーバーを再起動します。
- Nagios サーバーがインストールされているシステムに PEM エージェントをインストールし、PEM サーバーに登録します。そのエージェントの `agent.cfg` ファイルで `enable_nagios` 構成を `true` に設定し、エージェントサービスをリスタートします。

Nagios アラートを有効にするようにサーバーを構成すると、トリガーされたアラートはパッシブチェック結果を Nagios サービスに送信します。パッシブアラートの構文は次のとおりです。

```
<timestamp> PROCESS_SERVICE_CHECK_RESULT; <host_name> ; <service_name> ;
<service_status> ;
```

そこで

`timestamp` は、アラートがトリガーされた日時です。

`host_name` は、サーバーまたはエージェントの名前です。

`service_name` はアラートの名前です。

`service_status` は、サービスステータスの数値です。

- サービスステータスが OK の場合は 0
- サービスステータスが WARNING の場合 1
- サービスステータスが CRITICAL の場合 2
- 3 サービスステータスが UNKNOWN の場合

PEM サーバーは、次のルールを使用してサービスステータスを評価します。

- PEM 警告レベルが CLEARED の場合、警告メッセージは OK となります。
- PEM 警告レベルが低い場合、警告メッセージは WARNING となります。
- `is_nagios_medium_alert_as_critical` フラグ PEM サーバー構成ダイアログボックスで指定が FALSE に設定され、アラートレベル MEDIUM の場合、警告メッセージは WARNING となります。
- `is_nagios_medium_alert_as_critical` フラグ PEM サーバー構成ダイアログボックスで指定が TRUE に設定され、アラートレベルが MEDIUM の場合、警告メッセージは CRITICAL となります。
- PEM 警告レベルが HIGH の場合、警告メッセージは CRITICAL となります。

### アラートの Nagios 通知を有効にする

PEM サーバーは、有効なアラートごとに一意の通知プロパティセットを維持します。**Manage Alerts** タブの **Notification** タブを使用して、トリガーされたときに、特定のアラートが Nagios にアラート通知を送信するように指定します。

アラートの通知プロパティを変更するには、アラートによって監視されるオブジェクトの名前を右クリックし、**Management > Alerts > Manage Alerts** を選択します。 [ **Manage Alerts** ] タブで、Alerts リストでアラート名の左側にある **Edit** を選択します。編集ページが開いたら、[ **Notification** ] タブを選択します。

Nagios 通知を有効にするには、**Nagios へのパッシブサービスチェック結果の送信**の横にスライダーを移動し、 **Yes** をクリックします。次に、 **Save** を選択します。

### PEM サーバーの Nagios 関連動作の構成

[ Server Configuration ] ダイアログボックスを使用して、Nagios 構成に関する情報を PEM サーバーに提供できます。ダイアログボックスを開くには、**Management > Server Configuration** を選択します。

4 つのサーバー構成パラメーターは、Nagios インストールと Nagios に関連する PEM サーバーの動作に関する情報を指定します。

- `nagios_cmd_file_name` パラメーターを使用して、PEM からパッシブチェックアラートを受信する Nagios パイプラインファイルの場所を指定します。このパラメーターのデフォルト値は`/usr/local/nagios/var/rw/nagios.cmd` です。 `nagios.cmd` ファイルが他の場所にある場合、 **Value** フィールドにファイルの場所を指定します。
- `nagios_enabled` パラメーターのスライダーを **Yes** に移動して、パッシブチェックアラートを Nagios に送信するように PEM サーバーに指示します。
- `nagios_medium_alert_as_critical` スライダーを使用して、中アラートがトリガーされた場合に PEM サーバーが Nagios に渡す警告の重大度を指定します。
  - `is_nagios_medium_alert_as_critical` フラグが **FALSE** に設定され、アラートレベルが **MEDIUM** の場合、警告メッセージは WARNING.
  - `is_nagios_medium_alert_as_critical` フラグが **TRUE** に設定され、アラートレベルが **MEDIUM** の場合、警告メッセージは CRITICAL となります。
- `nagios_spool_retention_time` パラメーターを使用して、PEM サーバーに保存する通知履歴の日数を指定します。デフォルト値は 7 日です。

パラメーター値を変更した後、ダイアログボックスの右上隅にある **Save** を選択します。

### hosts.cfg および services.cfg ファイルの作成

`templates.cfg` ファイルデフォルトでは`/usr/local/nagios/etc/objects` にありますは、汎用ホストと汎用サービスのプロパティを指定します。プロパティは、`hosts.cfg` および `services.cfg` ファイルで使用されるパラメーターを指定します。

ほとんどの場合、PEM がデフォルト構成でインストールされている場合、`hosts.cfg` および `services.cfg` ファイルを作成する前に `templates.cfg` ファイルを変更する必要はありません。必要に応じて、`templates.cfg` ファイルを変更して、パラメーターの代替値を指定したり、新しいテンプレートを作成したりできます。

Nagios 構成ファイルを変更する前に、次のコマンドを使用して、ローカルシステムにある PEM ホストに関する情報を含む hosts.cfg ファイルを作成します。

```
psql -U postgres -p 5433 -d pem -A -t -c "select pem.create_nagios_host_config('generic-host')"
> /usr/local/nagios/etc/objects/hosts.cfg
```

次に、次のコマンドを使用して、ローカルシステムにある PEM サービスに関する情報を含む services.cfg ファイルを作成します。

```
psql -U postgres -p 5433 -d pem -A -t -c "select pem.create_nagios_service_config('generic-service')"
> /usr/local/nagios/etc/objects/services.cfg
```

custom template.cfg ファイルエントリーを使用する場合は、これらのコマンドで generic-host または generic-service の代わりにエントリー名を指定します。

#### Nagios 構成ファイルの変更

host.cfg および services.cfg ファイルを作成した後、Nagios 構成ファイルデフォルトでは/usr/local/nagios/etc/nagios.cfg でそれらの場所を指定する必要があります。構成ファイルを変更し、ファイルの場所を指定するエントリを追加します。

```
cfg_file=/usr/local/etc/objects/hosts.cfg
```

```
cfg_file=/usr/local/etc/objects/services.cfg
```

次のコマンドを使用して、Nagios が適切に構成されていることを確認できます。

```
/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg
```

Nagios が正しく構成されていることを確認した後、Nagios サービスをリスタートします。

```
/usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
```

### 8.3.5 Capacity Manager

PEM の Capacity Manager は、収集された統計メトリックを分析して、オブジェクトの履歴使用統計を表示するグラフまたは表を生成します。オブジェクトの予想使用統計を投影することもできます。特定のホスト、サーバー、データベース、またはデータベースオブジェクトのメトリックを収集および分析するように Capacity Manager を構成できます。

以下を選択して、Capacity Manager レポートの内容を調整できます。

- レポートに含める特定のメトリック
- メトリックが収集された時間範囲
- 分析されるメトリックの高しきい値または低しきい値

Capacity Manager レポートの開始日と終了日を指定することもできます。レポートの終了日が将来の時刻を指定している場合、Capacity Manager は選択されたオブジェクトの使用履歴を分析し、将来の予測オブジェクト使用量を推定します。

Capacity Manager を開くには、PEM クライアントで、 **Management > Capacity Manager** を選択します。Capacity Manager ウィザードが開き、[ **メトリック** ] タブにツリーが表示されます。 [ **Metrics** ] タブで、ツリーを展開して、分析するノードのメトリックを確認します。レポートに含めるメトリックを選択します。

Capacity Manager は、 **Aggregation** リストで指定された集計方法を使用します。 Capacity Manager は、集約方法を使用してメトリック値を評価およびプロットします。以下から選択してください

- **Average** - 期間中に記録された値の平均を使用します。
- **最大値** - 期間中に記録された最大値を使用します。
- **最小** - 期間中に記録された最小値を使用します。
- **First** - 期間中に記録された最初の値を使用します。

Capacity Manager レポートからメトリックを削除するには、メトリックの名前の左側にあるボックスをクリアします。

**Graph/chart metrics individually?** の横にスライダーを移動し**をクリックして、** メトリック**タブで選択した各メトリックの個別のレポートを生成します。オプションが** No**に設定されている場合、選択されたすべてのメトリックが単一のグラフまたは表にマージされます。

**Generate** を選択してレポートを画面に表示するデフォルトの構成オプションを受け入れますか、**オプション**タブを使用してサンプリング境界、レポートタイプ、レポートの宛先をカスタマイズします。[ **オプション**] タブに表示される時間は、PEM クライアントが存在するタイムゾーンからのものです。

**Time Period** ボックスのフィールドを使用して、Capacity Manager レポートの境界を定義します。

- **Period** リストを使用して、レポートに使用する期間の種類を選択します。以下を選択できます。

| Start time and end time | Specify a start date and an end date/time for the report |
|---|---|
| Start time and threshold | Specify a start date and time and a threshold to determine the end time and date for the report. |
| Historical days and extrapolated days | Specify a start date for the report that is a number of days in the past and an end date that is a number of days in the future. This option is useful for report templates that don't specify fixed dates. |
| Historical days and threshold | Specify a start date that is a number of days in the past, and end it when a threshold value is reached. |

レポートの期間の種類を指定した後、 **期間**ボックスの他のオプションから選択してレポートの期間を定義します。

- **開始時刻**フィールドの横にある日付と時刻セレクターを使用して、サンプリング期間の開始日時を指定します。または、レポートに含めるデータの履歴日数を選択します。**開始時刻**フィールドで指定された日付と時刻は、現在の日付/時刻より後であってはなりません。

デフォルトでは、Capacity Manager は現在の日時の 1 週間前の開始時刻を選択します。

- レポートの終了境界は、時間、将来の日数、または選択したメトリックが指定されたしきい値に到達したポイントです。**終了時間**の横にある日付と時刻セレクターを使用して、レポートの終了境界を指定します。または、レポートに含めるデータの推定日数を選択します。**終了時間** フィールドで指定された時間は、 **開始時間**フィールドで指定された時間よりも後でなければなりません。

将来の終了日時を選択した場合、Capacity Manager は使用履歴情報を使用して、予想される将来の使用量を推定します。予想される使用量は履歴データのサンプリングにもとづいていますので、サンプリング期間が長くなると、将来の使用傾向の精度が向上します。

しきい値を指定するには、 **Threshold** フィールドのリストを使用して、メトリック、演算子 **Exceeds** または **Falls Below** 、およびメトリックのターゲット値を選択します。しきい値を使用してレポートの終了を定義する場合、Capacity Manager レポートは、選択したメトリックの値が指定された値を超えるまたは下回ると終了します。

`cm_max_end_date_in_years` 構成パラメーターは、Capacity Manager レポートの終了境界のデフォルト時間値を定義します。レポートの終了境界としてしきい値を指定し、最大時間が経過する前に境界の予想使用量が満たされない場合、レポートは `cm_max_date_in_years` パラメーターで指定された時間に終了します。デフォルトでは、`cm_max_end_date_in_years` は 5 です。[サーバーの構成] ダイアログボックスを使用して、`cm_max_end_date_in_years` の値を変更できます。

**Report** ボックスのフィールドは、レポート種類と宛先を指定します。**含めるレポート** オプションを使用して、Capacity Manager によって生成されるレポートの種類を指定します。

- PEM クライアントウィンドウに折れ線グラフの形式でレポートを表示するには、**Graph** を選択します。
- **Table of data** を選択して、レポートデータを含む表を PEM クライアントウィンドウに表示します。
- PEM クライアントウィンドウに折れ線グラフとデータテーブルの両方を表示するには、**Graph and table of data** を選択します。

**レポートの宛先**オプションを使用して、レポートを表示または保存する場所を設定します。

- PEM クライアントの新しいタブにレポートを表示するには、**New tab** を選択します。 Capacity Manager レポートの第 1 世代を表示するには、**新しいタブ** を選択する必要があります。後続のレポートでは、**前のタブ** を選択できます。
- レポートを表示するときに以前に開いたタブを再利用するには、**前のタブ** を選択します。
- **Download the Report as a file** を選択し、ファイル名を指定して、指定したファイルにレポートを書き込みます。
- **Report Type?** で、 **HTML** または **JSON** タイプバージョン 9.6 以降を選択します。

レポート境界を指定し、Capacity Manager レポートの種類と宛先を選択した後、**Generate** を選択してレポートを作成します。

Capacity Vector HTML レポートは、スケーラブルベクターグラフィック SVG をサポートする Web ブラウザーで確認できます。SVG をサポートしていないブラウザーは Capacity Manager グラフを表示できません。そして不要な文字が含まれる場合があります。

#### 8.3.5.1 Capacity Manager テンプレート

レポートを定義したら、今後のレポートのテンプレートとして定義を保存できます。すべての PEM ユーザーは、Capacity Manager レポートテンプレートにアクセスできます。レポート定義をテンプレートとして保存するには

1. **Metrics** および **Options** タブを使用して、レポートを定義します。
2. **Save** を選択します。
3. **Save Template** ダイアログボックスの **Title** フィールドにレポート名を入力し、ツリー内のテンプレートを保存する場所を選択します。

4. **OK** を選択します。

レポートを作成するときに、**Load Template の Load Template** を使用して、既存のテンプレートを参照して開くことができます。開いたら、レポート定義を変更し、新しいテンプレートとして、または元のテンプレートを上書きして、再度保存できます。

**Manage Templates** を選択して、不要なテンプレートの名前を変更したり削除したりできるダイアログボックスを開きます。

#### 8.3.5.2 Capacity Manager メトリック - リファレンス

使用可能な Capacity Manager メトリックはプラットフォームによって異なり、変更される場合があります。使用可能なメトリックには、表で説明されているメトリックを含めることができます。

| Metric name | Description |
|---|---|
| # Dead Tuples | 選択したテーブルのデッドタプルの数。 |
| # Dead Tuples+ | 選択したテーブルのデッドタプルの累積数。 |
| # Heap Tuples Fetched by Index Scans | インデックススキャンによって取得されたヒープタプルの数。 |
| # Heap Tuples Fetched by Index Scans | インデックススキャンによって取得されたヒープタプルの累積数。 |
| # Idle Backends+ | 現在アイドル状態のバックエンドクライアントの累積数。 |
| # Index Scans | 指定されたオブジェクトで実行されたインデックススキャンの数。 |
| # Index Scans+ | 指定されたオブジェクトで実行されたインデックススキャンの累積数。 |
| # Index Tuples Read | 読み取られたインデックスタプルの数。 |
| # Index Tuples Read+ | 読み取られたインデックスタプルの累積数。 |
| # Live Tuples | トランザクションに可視されるタプルの数。 |
| # Live Tuples+ | トランザクションに表示されるタプルの累積数。 |
| # Pages Estimated by ANALYZE | ANALYZE によって推定されたページ数。 |
| # Pages Estimated by ANALYZE+ | ANALYZE によって推定された累積ページ数。 |
| # Sequential Scans | 特定のテーブルで実行されたシークエンシャル スキャンの数。 |
| # Sequential Scans+ | 特定のテーブルで実行されたシークエンシャル スキャンの累積数。 |
| # Sequential Scan Tuples | 特定のテーブルで順次スキャンされたタプルの数。 |
| # Sequential Scan Tuples+ | 特定のテーブルで連続してスキャンされたタプルの累積数。 |
| # Tuples Deleted | 削除されたタプルの数。 |
| # Tuples Deleted+ | 削除されたタプルの累積数。 |
| # Tuples Estimated by ANALYZE | ANALYZE によって推定されたライブ可視タプルの数。 |
| # Tuples Estimated by ANALYZE+ | ANALYZE によって推定されたライブタプルの累積数。 |
| # Tuples HOT Updated | タプル HOT の数が更新されました。 HOT 更新では、新しいタプルは元のタプルと同じブロックにあり、タプルはインデックスエントリを共有します。 |
| # Tuples HOT Updated+ | 更新されたタプル HOT の累積数。 |

次のページに続く

表 36 – 前のページからの続き

| Metric name | Description |
|---|---|
| # Tuples Inserted | 指定されたテーブルに挿入されたタプルの数。 |
| # Tuples Inserted+ | 指定されたテーブルに挿入されたタプルの累積数。 |
| # Tuples Updated | 選択したテーブルで更新されたタプルの数。 |
| # Tuples Updated+ | 選択したテーブルで更新されたタプルの累積数。 |
| Blocks Hit | キャッシュで見つかったブロックの数。 |
| Blocks Hit+ | キャッシュで見つかったブロックの累積数。 |
| Blocks Read | 読み取られたブロックの数。 |
| Blocks Read+ | 読み取られたブロックの累積数。 |
| Blocks Read from InfiniteCache | InfiniteCache から読み取られたブロックの数。 |
| Blocks Read from InfiniteCache+ | InfiniteCache から読み取られたブロックの累積数。 |
| Blocks Written | 書き込まれたブロックの数。 |
| Blocks Written+ | 書き込まれたブロックの累積数。 |
| Buffers Allocated | 割り当てられたバッファの数。 |
| Buffers Allocated+ | 割り当てられたバッファの累積数。 |
| Buffers Written - Backends | サーバープロセスクライアントアプリケーションに接続されているプロセスごとにディスクに書き込まれたバッファブロックの数。 |
| Buffers Written - Backends+ | サーバープロセスによってディスクに書き込まれたバッファブロックの累積数。 |
| Buffers Written - Checkpoint | チェックポイントプロセスによってディスクに書き込まれたブロック数。 |
| Buffers Written - Checkpoint+ | チェックポイントプロセスによってディスクに書き込まれた累積ブロック数。 |
| Buffers Written - Cleaning Scan | 自動バキュームプロセスによってディスクに書き込まれたブロック数。 |
| Buffers Written - Cleaning Scan+ | 自動バキュームプロセスによってディスクに書き込まれた累積ブロック数。 |
| Bytes Received (KB) | クライアントから受信したバイト数キロバイト単位。 |
| Bytes Received (KB)+ | 累積受信バイト数キロバイト単位。 |
| Bytes Sent (KB) | クライアントに送信されたバイト数キロバイト単位。 |
| Bytes Sent (KB)+ | 累積送信バイト数キロバイト単位。 |
| Checkpoints - Timed | チェックポイント間隔によってトリガーされたチェックポイント操作の数。 |
| Checkpoints - Timed+ | チェックポイント間隔によってトリガーされたチェックポイント操作の累積数。 |
| Checkpoints - Untimed | チェックポイントサイズによってトリガーされたチェックポイント操作の数。 |
| Checkpoints - Untimed+ | チェックポイントサイズによってトリガーされたチェックポイント操作の累積数。 |
| Database Size (MB) | 指定されたデータベースのサイズメガバイト単位。 |
| Free RAM Memory | 空き RAM メモリの量メガバイト単位。 |
| Free Swap Memory | ディスク上の空きスワップ領域の量メガバイト単位。 |
| Heap Blocks Hit | キャッシュで見つかったヒープブロックの数。 |
| Heap Blocks Hit+ | キャッシュで見つかったヒープブロックの累積数。 |

次のページに続く

表 36 – 前のページからの続き

| Metric name | Description |
|---|---|
| Heap Blocks Read | 読み取られたヒープブロックの数。 |
| Heap Blocks Read+ | 読み取られたヒープブロックの累積数。 |
| Index Blocks Hit | キャッシュで見つかったインデックスブロックの数。 |
| Index Blocks Hit+ | キャッシュで見つかったインデックスブロックの累積数。 |
| Index Blocks Read | 読み取られたインデックスブロックの数。 |
| Index Blocks Read+ | 読み取られたインデックスブロックの累積数。 |
| Index Size (MB) | 指定されたインデックスのサイズメガバイト単位。 |
| In Packets Discards | 破棄されたインバウンドパケットの数。 |
| In Packets Discards+ | 破棄されたインバウンドパケットの累積数。 |
| In Packets Errors | エラーを含む受信パケットの数。 |
| In Packets Errors+ | エラーを含む受信パケットの累積数。 |
| Link Bandwidth (Mbit/s) | ネットワークアダプターの速度メガビット/秒。 |
| Load Average - 15 Minute | CPU 飽和パーセント単位-15 分のサンプリング平均。 |
| Load Average - 1 Minute | CPU 飽和パーセント単位-1 分間のサンプリング平均。 |
| Load Average - 5 Minute | CPU 飽和パーセント単位-5 分間のサンプリング平均。 |
| Load Percentage | パーセント単位の CPU 飽和度。 |
| Number of Prepared Transactions+ | 準備されたトランザクションの累積数。 |
| Number of WAL Files+ | ログ先行書き込みファイルの累積数。 |
| Out Packets Discards | 破棄された送信パケットの数。 |
| Out Packets Discards+ | 破棄された送信パケットの累積数。 |
| Out Packets Errors | エラーを含む送信パケットの数。 |
| Out Packets Errors+ | エラーを含む送信パケットの累積数。 |
| Packets Received | 受信したパケット数。 |
| Packets Received+ | 受信したパケットの累積数。 |
| Packets Sent | 送信されたパケット数。 |
| Packets Sent+ | 送信されたパケットの累積数。 |
| Size (MB) | ディスクの合計サイズメガバイト単位。 |
| Size of Indexes (MB) | 指定されたテーブルのインデックスのサイズメガバイト単位。 |
| Space Available (MB) | 現在使用可能なディスク領域メガバイト単位。 |
| Space Used (MB) | 現在使用されているディスク領域メガバイト単位。 |
| Table Size (MB) | 指定されたテーブルのサイズメガバイト単位。 |
| Tablespace Size (MB) | 指定されたテーブルスペースのサイズメガバイト単位。 |
| Temp Buffers (MB) | 一時バッファのサイズメガバイト単位。 |
| Toast Blocks Hit | キャッシュで見つかった TOAST ブロックの数。 |
| Toast Blocks Hit+ | キャッシュで見つかった TOAST ブロックの累積数。 |
| Toast Blocks Read | 読み取られた TOAST ブロックの数。 |
| Toast Blocks Read+ | 読み取られた TOAST ブロックの累積数。 |
| Total RAM Memory | システム上の RAM メモリの合計量メガバイト単位。 |
| Total Swap Memory | システム上のスワップ領域の合計量メガバイト単位。 |
| Total Table Size w/Indexes and Toast | 指定されたテーブルの合計サイズインデックスおよび関連する特大属性を含む |

次のページに続く

表 36 – 前のページからの続き

| Metric name | Description |
|---|---|
| Transactions Aborted | 中止されたトランザクションの数。 |
| Transactions Aborted+ | 中止されたトランザクションの累積数。 |
| Transactions Committed | コミットされたトランザクションの数。 |
| Transactions Committed+ | コミットされたトランザクションの累積数。 |
| Tuples Deleted | 指定されたテーブルから削除されたタプルの数。 |
| Tuples Deleted+ | 指定されたテーブルから削除されたタプルの累積数。 |
| Tuples Estimated by ANALYZE | 指定されたテーブルにある可視のタプルの数。 |
| Tuples Estimated by ANALYZE+ | 指定されたテーブル内の可視タプルの累積数。 |
| Tuples Fetched | 指定されたテーブルから取得されたタプルの数。 |
| Tuples Fetched+ | 指定されたテーブルから取得されたタプルの累積数。 |
| Tuples HOT Updated | タプル HOT の数が更新されました。 HOT 更新では、新しいタプルは元のタプルと同じブロックにあり、タプルはインデックスエントリを共有します。 |
| Tuples HOT Updated+ | 更新されたタプル HOT の累積数。 HOT 更新では、新しいタプルは元のタプルと同じブロックにあり、タプルはインデックスエントリを共有します。 |
| Tuples Inserted | 指定されたテーブルに挿入されたタプルの数。 |
| Tuples Inserted+ | 指定されたテーブルに挿入されたタプルの累積数。 |
| Tuples Returned | 結果セットで返されたタプルの数。 |
| Tuples Returned+ | 結果セットで返されたタプルの累積数。 |
| Tuples Updated | 指定されたテーブルで更新されたタプルの数。 |
| Tuples Updated+ | 指定されたテーブルで更新されたタプルの累積数。 |
| WAL Segment Size (MB) | ログ先行書き込みのセグメントサイズメガバイト単位。 |

**注釈**

メトリックの名前の後にある+は、メトリックのデータが累積的に収集されることを意味します。 '+' 記号が続いていないメトリックは、ポイントインタイム値として収集されます。

### 8.3.6 Audit Manager

PEM Audit Manager を使用して、 EDB Postgres Advanced Server インスタンスの監査ログ構成を簡素化できます。 Audit Manager を使用すると、次のようなロギング属性を構成できます。

- PEM がログファイルを収集する頻度
- ログファイルに含まれるデータベースアクティビティの種類
- ログファイルをローテーションする頻度とタイミング

監査ログには、次のアクティビティが含まれる場合があります。

- データベースインスタンスに対して行われたすべての接続

- 失敗した接続試行
- データベースインスタンスからの切断
- すべてのクエリー SELECT ステートメント
- すべての DML ステートメント INSERT、UPDATE、DELETE
- すべての DDL ステートメントたとえば、 CREATE、DROP、ALTER

監査ログが PEM サーバーに保存されると、監査ログダッシュボードを使用して、読みやすい形式で情報を確認できます。Audit Log ダッシュボードでは、次の基準でログファイルをフィルタリングできます。

- タイムスタンプ範囲アクティビティが発生したとき
- アクティビティが発生したデータベース
- アクティビティを実行するユーザー
- 呼び出されるコマンドの種類

#### 8.3.6.1 EDB Postgres Advanced Server インスタンスサービス ID の設定

EDB Postgres Advanced Server インスタンスのロギングを構成するには、サーバーがバインドされたエージェントを備えた PEM 管理サーバーであり、サーバー登録にサービススクリプトの名前が含まれている必要があります。新しいサーバーを登録するときは、新しいサーバーダイアログボックスの **Advanced** タブの **サービス ID** フィールドにサービス名を含めます。

既存の登録および接続されたサーバーにサービス名を追加する前に、サーバーを切断する必要があります。

1. サーバー名を右クリックし、コンテキストメニューから **Disconnect server** を選択します。
2. サーバー名を右クリックし、コンテキストメニューから **Properties** を選択します。
3. **Advanced** タブを選択し、サービス名を **Service ID** フィールドに追加します。

サービス ID フィールドにより、PEM サーバーはサービスを停止および開始できます。

- Advanced Server 12 サービススクリプトの名前は `edb-as-12` です。
- Advanced Server 11 サービススクリプトの名前は `edb-as-11` です。
- Advanced Server 10 サービススクリプトの名前は `edb-as-10` です。

#### 8.3.6.2 EDB Audit Configuration プローブの設定

EDB Postgres Advanced Server サーバーの監査ログを設定する前に、EDB Audit Configuration プローブが有効になっていることを確認する必要があります。**Manage Probes** タブを開き、 Manage Probes のステータスを確認するには、ツリーで登録済みの EDB Postgres Advanced Server サーバーの名前を右クリックし、**Management > Manage Probes** を選択します。

EDB Audit Configuration probe Probe Configuration ダイアログボックスの Enabled 列が **Yes** に設定されていることを確認します。 EDB Audit Configuration が有効になっていない場合は、 **Manage Probes** タブの **Enabled?** スイッチを使用して有効にします。

#### 8.3.6.3 Audit Manager を使用した監査ログの構成

1. Audit Manager ウィザードを開くには、**Management > Audit Manager** を選択します。 Audit Manager - Welcome ダイアログボックスが開きます。**Next** を選択して続行します。

2. Select Servers ツリーを使用して、監査構成を適用するサーバーを指定します。ツリー内のサーバーを利用できるようにするには、PEM による監視のためのサーバーを登録するときに使う、Create Server ダイアログボックスの **Advanced** タブにサービス ID を提供しなければならない。 EDB Postgres Advanced Server のみが監査をサポートしています。PostgreSQL サーバーはツリーに表示されません。

**Next** を選択して続行します。

1. Auditing Parameters Configuration ダイアログボックスでは、監査を有効または無効にし、ログレコードが PEM に収集される頻度を選択できます。Auditing Parameters Configuration ダイアログボックスを使用して、監査設定を指定します。

   - **Auditing** スイッチを使用して、指定したサーバーで監査を有効または無効にします。
   - **Audit destination** リストを使用して、監査ログの宛先を選択します。 **ファイル** または **Syslog** を選択します。この機能は EDB Postgres Advanced Server 10 以降のリリースでのみサポートされています。
   - **Import logs to PEM** スイッチを使用して、各サーバーから PEM サーバーにログレコードを定期的にインポートします。ログファイルをインポートするには、スイッチを **Yes** に設定します。デフォルトは **No** です。
   - **Collection frequency** リストを使用して、ログ収集が有効になっているときに PEM が監視対象サーバーからログレコードを収集する Collection frequency **を指定します。

   -** Log format **リストを使用して、生のファイルを選択します各サーバーに書き込むログ形式。ログ収集が有効になっている場合、PEM サーバーは CSV 形式を使用します。

   -** File name**フィールドを使用して、ログファイル名を生成するときに使用される形式を指定します。デフォルトでは、形式は `audit-%Y-%m-%d_%H%M%S` に設定されます。ここで

`audit` は監査ディレクトリ名フィールドで指定されたファイル名です `Y` はログが保存された年 `m` はログが保存された月 `d` はログが保存された日 `H` はログが保存された時間 `M` はログが保存された分 `S` はログが保存された秒です b_tran_9 **Change Log Directory for selected servers?** を選択し、 **Audit Directory Name** フィールドを使用して、監査を書き込むディレクトリ名を指定しますログ。ディレクトリは PEM サーバーのデータディレクトリの下にあります。

- **Log directory** ボックスを使用して、ログファイルが保存されるディレクトリに関する情報を指定します。

```
-   **Change log directory for selected servers?** を移動します **Directory name**␣
↪フィールドを有効にするには、   **Yes** に切り替えます。

-   **Directory name** フィールドを使用して、監査ログを書き込む各サーバーのディレクトリの名前を指定します。指定されたディレクトリは、サーバー上のデータディレクトリのサブディレクトリとして作成されます。
```

**Next** を選択して続行します。

1. Audit log configuration ダイアログボックスは、Auditing Parameters Configuration ダイアログボックスで監査を有効にしている場合にのみ使用できます。Audit Log Configuration ダイアログボックスを使用して、各サーバーに適用するログ構成の詳細を指定します。

    - **Connection attempts** スイッチを使用して、Connection attempts ログに記録するかどうかを指定します。以下を指定します

接続ログを無効にする **None**

**All** は、すべての接続試行がログに記録されることを示します

**Failed** は、ログ記録に失敗した接続試行のログ b_tran_1 **Disconnection attempts** スイッチを使用して、Disconnection attempts ログに記録するかどうかを指定します。以下を指定します

切断ログに記録したくない場合は **None**

切断ロギングを有効にするための **All** bb_tran_1 **Log attempts** フィールドを使用して、ログに記録されるステートメントタイプを指定します。以下から選択してください

**Select** - SELECT キーワードを含むすべてのステートメントをログに記録します。

**Error** - Error になったすべてのステートメントをログに記録します。

**DML** - すべての DML データ変更言語ステートメントをログに記録します。

**DDL** - すべての DDL データ定義言語ステートメント、つまりデータを追加、削除、または変更するステートメントをログに記録します。

**Select All** を選択して、すべてのステートメントタイプを選択します。

すべてのステートメントタイプをクリアするには、 **Unselect All** を選択します。

- **Audit tag** フィールドを使用して、収集されたログの追跡タグを指定します。監査タグ付け機能は、EDB Postgres Advanced Server のサポートされているバージョンでのみ利用できます。
- **Log rotation** ボックスを使用して、各サーバーでログファイルを管理する方法を指定します。

**Enable?** スイッチを使用して、ログファイルをローテーションします。新しいログファイルを定期的に使用して、単一のファイルが大きくなりすぎないようにします。

**Day** リストを使用して、ログファイルをローテーションする日を選択します。

**Size (MB)** フィールドを使用して、ログファイルをローテーションするサイズをメガバイト単位で指定します。

**Time (seconds)** フィールドを使用して、ログファイルのローテーション間の秒数を指定します。

**Next** を選択して続行します。

1. Schedule Auditing Changes ダイアログボックスを使用して、監査構成の変更を有効にする時期を決定します。

    - 監査構成の変更をすぐに実行する場合は、 **Configure locations now?** を選択します。影響を受けるデータベースサーバーが再起動するため、監査変更を有効にできます。
    - **Time?** セレクターを使用して、将来のある時点で監査構成の変更をスケジュールします。リストから目的の日付と時刻を選択します。影響を受けるデータベースサーバーは、指定された日時に再起動して、監査の変更を有効にします。

**Finish** を選択して、監査構成プロセスを完了します。

Audit Manager は、各サーバーに構成を適用するジョブをスケジュールします。ジョブは 2 つのタスクで構成されています。1 つはサーバー上の監査ログ構成を更新し、もう 1 つは新しい構成でサーバーを再起動します。

**Scheduled Tasks** タブを使用して、スケジュールされたジョブのリストを確認できます。**Scheduled Tasks** タブを開くには、サーバーまたはエージェントの名前を選択し、 **Management > Scheduled Tasks** を選択します。

#### 8.3.6.4 Audit Log ダッシュボードでのログの表示

Audit Log ダッシュボードを使用して、 EDB Postgres Advanced Server データベースインスタンスから audit log を表示します。

Audit Log ダッシュボードを開くには、サーバーまたはエージェントノードを右クリックし、**Dashboards** メニューから **Audit Log Analysis** を選択します。 Audit Log ダッシュボードには、監査レコードが新しい順に表示されます。

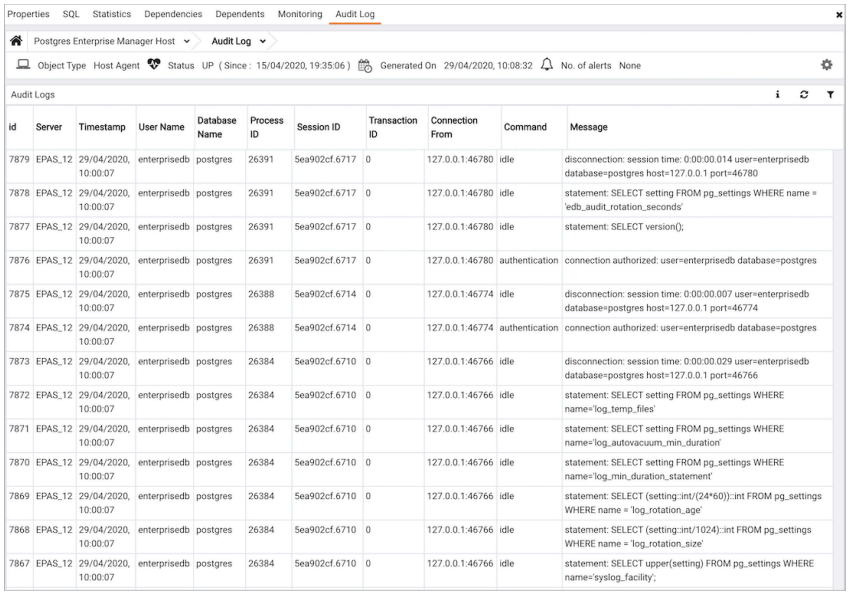



図 10: The Audit Log dashboard

フィルタリングを使用して、表示される監査レコードの数を制限できます。**Show Filters** を選択してフィルタパネルを公開します。Filters パネルを使用して、表示する監査レコードの選択基準を提供します。

- **Start** フィールドを使用して、レポートの開始日を指定します。
- **End** フィールドを使用して、レポートの終了日を指定します。
- **User** フィールドを使用して、特定の Postgres ユーザーによってアクティビティが開始されたエントリのみを表示します。
- **Server** フィールドを使用して、特定のデータベースでアクティビティが発行されたエントリのみを表示します。

- **Command type** フィールドを使用して、アクティビティが指定されたタイプのエントリのみを表示します。指定できるコマンドタイプは **idle** 、 **authentication** 、および **SELECT** です。ユーザアプリケーションから SQL ステートメントを表示する場合、idle コマンドタイプを指定します。

**Filter** を選択して、フィルター項目をログ項目に適用します。

### 8.3.7 Log Manager

PEM Log Manager を使用して、Postgres インスタンスのサーバーログ構成を簡素化できます。ログマネージャーを使用すると、次を含むすべてのサーバーログパラメーターを変更できます。

- ログファイルが書き込まれる場所
- ログファイルが書き込まれる頻度
- ログファイルに書き込まれる情報の種類
- ログファイルエントリのフォーマット
- ログローテーションのプロパティ

Log Manager を使用してサーバーのロギングプロパティを定義する前に、新しいサーバー登録またはプロパティの **Advanced** タブの **Service ID** フィールドに、関連する Advanced Server または PostgreSQL データベースサーバーの名前を指定する必要がありますダイアログボックス。サーバーは、**Service ID** フィールドでサービスの名前を指定した場合にのみ、Server Selection ダイアログボックスの構成に使用できます。

たとえば、Linux ホストにある Advanced Server 9.4 インスタンスのロギング設定を設定しているとします。監視対象サーバーの **Properties** ダイアログボックスの **Advanced** タブの **Service ID** フィールドを `ppas-9.4` に設定します。

1. Log Manager を開くには、PEM クライアントで、 **Management > Log Manager** を選択します。ウィザードが開き、Welcome メッセージが表示されます。**Next** を選択します。
2. Server Selection ダイアログボックスには、PEM によって監視されているサーバー接続のリストが表示されます。Log Manager ウィザードで指定した構成を適用するサーバーの名前の横にあるチェックボックスをオンにします。Log Manager は、サーバー選択ツリーで名前の左側に赤い感嘆符が表示されているサーバーでは無効になっています。サーバーはいくつかの理由で有効にされない場合があります。
   - Log Manager は、[ Properties ダイアログボックスの **Advanced** ] タブでサービス ID を指定するサーバーのみを構成できます。サービス ID を提供するには
   1. 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 **Disconnect Server** を選択します。
   2. プロンプトが表示されたら、パスワードを入力します。
   3. サーバーのコンテキストメニューから **を選択しますプロパティ** .
   4. **Advanced** タブで、 **Service ID** フィールドにサービスの名前を入力します。
   5. **Save** を選択します。
   - サーバーにバインドされた PEM エージェントがサーバーを再起動するための十分な特権がある場合、サーバーは無効になっています。

- サーバーにバインドされた PEM エージェントが、関連付けられた PEM サーバーよりも古いバージョンの場合、サーバーは無効になっています。

**Next** を選択します。

3. Log configuration ダイアログボックスを使用して、ログファイルを PEM にインポートする頻度を指定し、ログローテーションの詳細を指定します。

**Import Logs** ボックスのオプションは、ログファイルを PEM にインポートする頻度を指定します。

- **Import logs to PEM** の横にあるスイッチを使用して、ログファイルを PEM にインポートし、それらを表示するかどうかを指定します the Server Log Analysisashboard.
- **Import Frequency** リストを使用して、ログファイルを PEM にインポートする頻度を指定します。

**Log rotation configuration** ボックスを使用して、ログファイルの最大長ライフスパンまたはサイズを指定します。

- **Rotation Size** フィールドを使用して、ログファイルの最大サイズをメガバイト単位で指定します。デフォルト値は 10MB です。ログファイルの最大サイズに制限を設けたくない場合は、0 に設定します。
- **Rotation Time** フィールドを使用して、各ログファイルに保存する全体の日数を指定します。デフォルト値は 1 日です。

**Truncation on Rotation** スイッチを使用して、時間ベースのログファイルローテーションのサーバー動作を指定します。

- **On** を選択して、新しいファイルと同じ名前の既存のログファイルを上書きします。
- **Off を選択します**新しいログファイルと同じ名前の既存のログファイルに、新しいログファイルエントリを追加します。これはデフォルトの動作です。

**Next** を選択します。

4. Where to Log ダイアログボックスを使用して、ログファイルを書き込む場所を指定します。

    - **Log Destination** ボックスからオプションを選択して、サーバーログ出力の宛先を指定します。
    - **stderr** スイッチを次のように設定します。コンマ区切り値形式でログファイルを書き込むには、**csvlog** スイッチを **Yes** に設定します。このオプションは、[ Schedule ] ダイアログボックスで **Import logs to PEM** を選択した場合、有効になり編集できません。サーバーログファイルを PEM にインポートしない場合は、このオプションを編集できます。
    - ログファイルをシステムログファイルに書き込むには、 **syslog** スイッチを **Yes** に設定します。
    - Windows では、**を設定しますログファイルをイベントログに書き込むには、**eventlog** スイッチを **Yes** にします。
    - **Log collection** ボックスを使用して、収集設定を指定します。
    - **Log Collector** スイッチを **Enable に設定します**キャプチャされたログメッセージ stderr にリダイレクトされます。
    - **Log Silent Mode** スイッチを **Enable** に設定して、制御端末からの関連付けを解除して、バックグラウンドでサーバーをサイレント実行します。
    - **Log Directory** ボックスを使用して、ログファイルの場所設定を指定します。

- ログファイルの各セットを別のディレクトリに維持するには、**Change log directory for selected servers?** スイッチを **Yes** に設定します。
- **ディレクトリ名**フィールドを使用して、ログファイルを書き込むディレクトリを指定します。ディレクトリは、監視対象サーバーのインストールディレクトリの下の pg_log ディレクトリの下にあります。
- **Log File Name** フィールドを使用して、ログファイル名の形式を指定します。**デフォルト**に設定されている場合、フォーマットは enterprisedb-%Y-%m-%d_%H%M%S であり、ここで
- enterprisedb はファイル名接頭辞です b_tran_31 Y はログが保存された年です b_tran_31 m はログが保存された月です b_tran_31 d はログが保存された日です b_31 H は、ログが保存された時間です
- M は、ログが保存された分です
- S は、ログが保存された秒です b_tran_29 syslog へのロギングが有効な場合
- **Syslog Facility** リストを使用して、syslog 機能を指定します use.
- **Syslog Ident** フィールドを使用して、システムログの EDB Postgres Advanced Server エントリを識別するプログラム名を指定します。

**Next** を選択します。

5. When to Log ダイアログボックスを使用して、ログファイルエントリを開始するイベントを指定します。
   - 重大度の順に重大度レベルは次のとおりです。
   - **panic** - すべてのデータベースを発生させたエラー servers to abort.
   - **fatal** - セッションを中止するエラー。
   - **log** - 管理者が関心のある情報メッセージ。
   - **error** - コマンドを中止するエラー。
   - **warning** - コマンドは完了したが、予想通りに実行されないエラー条件。
   - **notice** - ユーザーが関心のある項目。これはデフォルトです。
   - **info** - ユーザーが暗黙的に要求した情報。
   - **debug5** から **debug1** - 開発者に役立つ詳細なデバッグ情報。
   - **Client min attempts** リストを使用する
   - **Log min attempts** リストを使用して、サーバーログに書き込む最も低い重大度レベルを指定します。
   - デフォルトでは、エラーメッセージがサーバーログには、ログエントリを開始した SQL ステートメントのテキストは含まれません。**Log min error Statement** リストを使用して、SQL ステートメントロギングをトリガーする重大度レベルを指定します。メッセージが指定された重大度以上の場合、メッセージを生成した SQL ステートメントはサーバー log.
   - に書き込まれます。 **Log minduration Statement** リストを使用して、ミリ秒単位のステートメント期間を指定します。指定されたミリ秒数を超えるステートメントはサーバーログに書き込

まれます。値**-1** は、すべての期間ベースのロギングを無効にします。値 **0** は、すべてのステートメントとその期間をログに記録します。

- **Log temp files** フィールドを使用して、ファイルサイズをキロバイト単位で指定します。一時ファイルが指定されたサイズに達すると、ログに記録されます。この機能を無効にするには、**-1** デフォルトの値を指定します。
- **Log autoVacuum minduration** フィールドを使用して、ミリ秒単位の時間長を指定します。自動バキュームが指定された時間を超えると、アクティビティがログに記録されます。この機能を無効にするには、値**-1** デフォルトを指定します。

**Next** を選択します。

6. What to Log ダイアログボックスを使用して、デバッグと監査に役立つログエントリオプションを指定します。

**Debug options** ボックスのスイッチを使用して、クエリの実行に関連するログファイルに、開発者が関心を持つ可能性のある情報を含めます。

- ログファイルの構文解析ツリー。
- ログファイルにクエリリライター出力を含めるには、**Rewriter Output** スイッチを **Yes** に設定します。
- **Execution plan** スイッチを **Yes** に設定します実行された各クエリの実行プランをログファイルに含めます。

**Indent Debug Options Output in Log** スイッチが **Yes** に設定されている場合、サーバーはパースツリーエントリ、クエリリライターエントリ、またはクエリ実行プランエントリを含む各行をインデントします。インデントにより、生成されるログファイルが読みやすくなりますが、ログファイルが長くなります。

**General Options** ボックスのスイッチを使用して、ログファイルに監査情報を含めます。

- サーバーログにチェックポイントとリスタートポイントを含めるには、**Checkpoints** スイッチを **Yes** に設定します。
- **を設定します** Connections** スイッチを **Yes** に設定して、サーバーへの接続が試行された各接続と、サーバーログに正常に認証された接続を含めます。
- サーバーログを含めるには、**Disconnections** スイッチを **Yes** に設定しますセッション情報とセッション期間を提供する、終了したセッションのエントリー。
- **Duration** スイッチを **Yes** に設定して、サーバーにログに記録された各ステートメントの実行に必要な時間を含めます。log.
- **を設定します** Hostname** は **Yes** に切り替えて、各サーバーログエントリに IP アドレスとホスト名の両方を含めます。デフォルトでは、IP アドレスのみがログに記録されます。これにより、パフォーマンスのペナルティが発生する可能性があります。
- ロックを取得するために `deadlock_timeout` パラメーターで指定された時間より長く待機するセッションのログエントリを書き込むには、**Lock Waits** スイッチを **Yes** に設定します。これは、ロック待機がパフォーマンスの低下の原因であるかどうかを判断する場合に役立ちます。

**Error verbosity** リストを使用して、サーバーログの各エントリに書き込まれる詳細を指定します。

- **default** を選択して、各サーバーログエントリにエラーメッセージ、詳細、ヒント、クエリ、およびコンテキストを含めます。

- エラーメッセージのみをログに記録するには **terse** を選択します。
- エラーメッセージ、DETAIL、HINT、QUERY、および CONTEXT エラー情報、SQLSTATE エラーコードとソースコードファイル名、ファンクション名を含めるには **verbose** を選択します、およびエラーを生成した行番号。

**Prefix string** フィールドを使用して、各ログファイルエントリの先頭に書き込まれる printf スタイルの文字列を指定します。サポートされるオプションについては、Postgres core documentation の log_line_prefix ドキュメントを参照してください。

**Statements** リストを使用して、サーバーログに含める SQL ステートメントを指定します。

- SQL ステートメントのロギングを無効にするには **none** デフォルトを指定します。
- ログ ddl に指示するには **ddl** を指定します CREATE、ALTER、DROP などのデータ定義言語ステートメント。
- すべての ddl ステートメントと、INSERT、UPDATE、DELETE、TRUNCATE などのすべての dml データ変更言語ステートメントをログに記録するには **mod** を指定します。と COPY FROM.
- すべての SQL ステートメントをログに記録するには **all** を指定します。

**Next** を選択します。

7. Schedule Logging Changes ダイアログボックスを使用して、ロギングが構成変更を適用する時期を指定します。
    - **Configure locations now** スイッチを **Yes** に設定して、構成設定を有効にします。ログマネージャーウィザード.
    - が完了すると、サーバーが再起動します。設定とサーバーを再起動します。

ログマネージャーウィザードで指定された構成の変更を適用すると、サーバーが再起動し、ユーザーのデータベースサーバーの使用を一時的に中断します。

8. **Finish** を選択してウィザードを終了します。サーバーを再起動するか、スケジュールダイアログボックスで指定された時間にサーバーの再起動をスケジュールします。

#### 8.3.7.1 Server Log Analysis ダッシュボードの確認

ログマネージャーウィザードを起動し、ログファイルを PEM にインポートした後、Server Log Analysis ダッシュボードを使用して、選択したサーバーのログファイルを確認できます。Server Log Analysis ダッシュボードを開くには、PEM クライアントツリーで監視対象サーバーの名前を右クリックし、**Dashboards > Server Log Analysis** を選択します。

Server Log Analysis ダッシュボードのヘッダー情報には、サーバーが起動された日時、ページが最後に更新された日時、およびトリガーされたアラートの現在の数が表示されます。

Server Log テーブルのエントリは時系列で表示され、最新のログエントリが最初に表示されます。スクロールバーを使用して、ログエントリを移動したり、表示されていない列を表示したりします。

サーバーログテーブルの上部の見出しは、各列に格納されている情報を示します。列見出しにカーソルを合わせると、各列のコンテンツの説明を含むツールチップが表示されます。

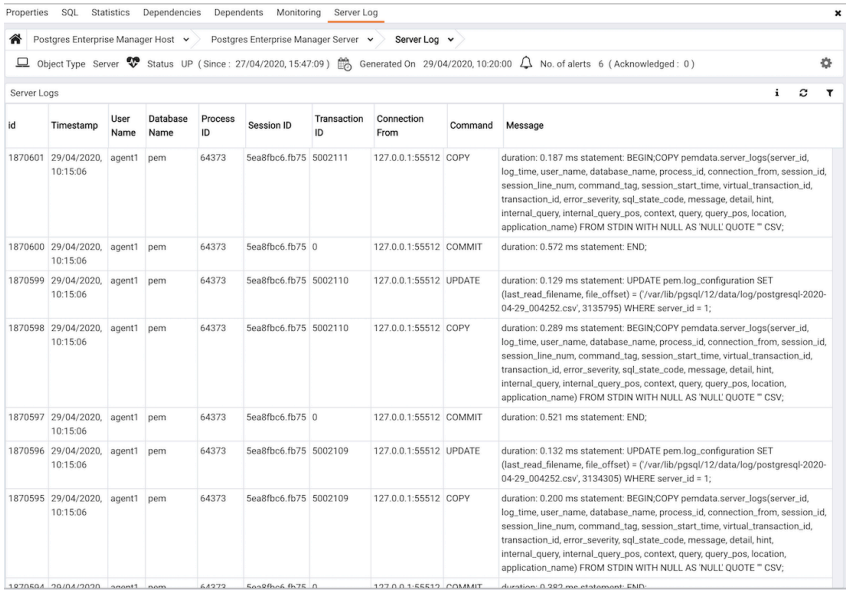



図 11: The Server Log Analysis dashboard

フィルタリングを使用して、表示されるサーバーログレコードの数を制限できます。**Show Filters** を選択してフィルタパネルを開き、フィルタを定義します。**フィルタ定義**ボックスを使用して、表示するレポートのサブセットを選択するための選択基準を説明します。

- **From** フィールドを使用して、表示されるサーバーログの開始日を指定します。
- **To** フィールドを使用して、表示されるサーバーログの終了日を指定します。
- **Username** フィールドにロール名を入力して、そのユーザーのみが実行したトランザクションを表示します。
- **Server** フィールドにデータベース名を入力して、表示されるレコードを指定したデータベースに対して実行されたトランザクションに制限します。
- **Command Type** フィールドを使用して、フィルタリングされたレポートに表示されるコマンドの選択基準を指定します。

サーバーログをフィルタリングするための基準を指定した後、 **Filter** を選択して、フィルタリングされたサーバーログを Server Log テーブルに表示します。

### 8.3.8 Charts

**Manage Charts** タブを使用して、カスタム折れ線グラフまたは表を作成または変更したり、カスタムチャートで使用する Capacity Manager テンプレートをインポートしたりできます。チャートを定義したら、カスタムダッシュボードにチャートを表示できます。

**Manage Charts** タブを開くには、PEM クライアントで、 **Management > Manage Charts** を選択します。[ **Manage Charts** ] タブは、次のことを行うダイアログボックスにアクセスするためのクイックリンクメニューを提供します。

- カスタムダッシュボードで使用する新しいチャートを作成します。

- Capacity Manager テンプレートをインポートして、カスタムチャートを作成するためのテンプレートとして使用します。

Custom Charts テーブルには、定義したチャートのリストが表示されます。チャートが新しい場合、フォントは緑色になります。グラフを追加するか、画面を更新すると、フォントが黒になります。Custom Charts テーブルの右上にある検索ボックスを使用して、カスタムチャートを検索します。以下を指定します

- チャート名
- 種類
- レベル
- メトリックカテゴリ

Custom Charts テーブルのチャート名の左側にあるアイコンを使用して、チャートを管理します。

- **Edit** を選択して Chart Configuration ウィザードを開き、チャートまたはテーブルの側面を変更します。
- 選択したチャートを削除するには、 **Delete** を選択します。

#### 8.3.8.1 カスタムチャートの作成

[ **Manage Charts** ] タブの [ Quick Links ] セクションで [ **Create New Chart** ] を選択して、Create Chart ウィザードを開きます。ウィザードは、新しいチャートを定義するために必要な手順をガイドします。

1. Configure Chart ダイアログボックスを使用して、チャートに関する一般情報を指定します。
    - **Name** フィールドにチャートの名前を指定します。
    - **Category** フィールドのリストを使用して、次のカテゴリを指定します。このチャートを表示するにはカスタムグラフをカスタムダッシュボードに追加する場合、指定したカテゴリからグラフを選択できます。
    - **Type** フィールドを使用して、グラフが折れ線グラフかテーブルかどうかを指定します。
    - でチャートの説明を提供します。 **Description** フィールド。説明は、ユーザーがチャートの横にある情報アイコンを選択すると、カスタムダッシュボードに表示されます。

[ Configure Chart ] ダイアログボックスのフィールドに入力したら、[ **Next** ] を選択します。

1. Select Metrics ダイアログボックスを使用して、チャートに表示するメトリックを選択します。
    - **メトリックレベル**リストを使用して、メトリックを選択する PEM 階層のレベルを指定します。 **エージェント** 、 **データベース** 、または **サーバー**を指定できます。各レベルは、プローブとメトリックの一意のセットへのアクセスを提供します。
    - **Available metrics** ボックスのツリーを使用して、チャートに表示するメトリックを選択します。

テーブルを作成する場合、1 つのプローブからのみメトリックを選択できます。ツリーの各ノードは、単一のプローブによって返されたメトリックをリストします。ツリーのノードを展開し、メトリック名の横にあるチェックボックスを選択して、そのメトリックデータをテーブルに含めます。

折れ線グラフを作成する場合は、ツリーのノードを展開し、チャートに含める各メトリックをダブルクリックします。

- Selected Metrics パネルを使用して、チャートにメトリックデータを表示する方法を指定します。選択パネルの変更不可の Metric [Probe] 列にメトリックの名前が表示されます。次のことができます。
  - ゴミ箱を選択して、選択したメトリックのリストからメトリックを削除します。
  - Selection Criteria 列のリストを使用して、表示されるデータの順序を指定します。
  - **Limit** フィールドを使用して数を指定するテーブルの行またはチャート内の行数最大 32 行 100 行。
- 折れ線グラフを作成している場合、PEM は階層間のメトリックの比較をサポートします。
  - **Compare** を選択して 1 つ以上を選択しますチャートで比較するプローブ固有の属性 CPU、インターフェイス、データベースなど。
  - **Copy** を選択して、同じプローブのすべてのメトリックに選択を適用します。

[ Select Metrics ] ダイアログボックスのフィールドに入力したら、[ **Next** ] を選択します。

1. Set Options ダイアログボックスを使用して、チャートの表示オプションを指定します。
   - **Auto Refresh** フィールドを使用して、チャートの更新の間分数を指定します。1〜120 の値を選択してください。デフォルトの自動リフレッシュレートは 2 分です。
   - **Line chart options** の下のフィールドを使用して、折れ線グラフの表示設定を指定します。
     - **プロットするポイント**フィールドを使用して指定しますチャートにプロットするポイントの最大数。
     - **履歴スパン**の右側のフィールドを使用して、チャートに日、時間、および分単位で表示する履歴データの量を指定します。
   - のフィールドを使用する **Data extrapolation** ボックスは、履歴データに基づいて PEM が推定データを生成するかどうかを指定します。
     - チャートから推定データを省略するには、 **No Extrapolation** を選択します。
     - **Days** を使用するには **Span** を選択します
     - しきい値セレクターを使用してチャートの最大値または最小値を指定するには、**Threshold** を選択します。

[ Set Options ] ダイアログボックスのフィールドに入力したら、[ **Next** ] を選択します。

1. Set Permissions ダイアログボックスを使用して、チャートの表示オプションを指定します。
   - 許可されたすべてのユーザーがチャートを利用できるようにするには、**Share with all** スライダーを **Yes** に設定します。**Accessmissions** フィールドで指定されたユーザーまたはグループへのアクセスを制限するには、**No** を選択します。
   - **Access attempts** フィールドを使用して、チャートへのアクセスを付与するグループを選択します。

チャートの定義が終了したら、 **Finish** を選択して編集を保存し、**Manage Charts** タブのリストにチャートを追加します。

#### 8.3.8.2 Capacity Manager テンプレートのインポート

**Manage Charts** タブの Quick Links セクションで **Import Capacity Manager Template** を選択して Create Chart ダイアログボックスを開き、Capacity Manager テンプレートをチャートまたはテーブルの開始点として使用します。

1. [ Create Chart ] ダイアログボックスが開いたら、カスタムチャートに関する情報を提供します。

    - **Import Capacity Template** リストを使用して、チャートのベースとなるテンプレートの名前を選択します。
    - チャートの名前を指定します **Name** フィールド.
    - **Category** リストを使用して、このチャートを表示するカテゴリを指定します。カスタムダッシュボードにカスタムチャートを追加する場合、このカテゴリーからチャートを選択できます。
    - **Type** フィールドを使用して、チャートが折れ線グラフまたはテーブルであることを指定します。
    - でチャートの説明を提供します。 **Description** フィールド。説明は、ユーザーがチャートの横にある情報アイコンを選択すると、カスタムダッシュボードに表示されます。

**Next** を選択します。

1. Select Metrics ウィンドウでは、選択したテンプレートで指定されたメトリックを確認できます。チャートエディターの下部パネルには、チャートに含めるメトリックが表示されます。チャートエディターを使用して、チャートに含まれるメトリックを変更することはできません。メトリックを変更するには、Capacity Manager ユーティリティを使用してテンプレートを更新します。

メトリックを確認したら、[ **Next** ] を選択します。

1. Set Options ウィンドウを使用して、チャートの表示オプションを指定します。

    - **Auto Refresh** フィールドを使用して、チャートの更新の間分数を指定します。1〜120 の値を選択してください。デフォルトの自動リフレッシュレートは 2 分です。
    - **Data extrapolation** ボックスを使用して、チャートの対象とする期間を指定します。次のいずれかができます。
    - **履歴日と外挿日** を選択し、以下を提供します。
        - **Historical** フィールドでグラフ化する履歴データの日数。
        - **Extrapolated でグラフ化する予測日数** field.
    - **履歴の日数としきい値** を選択して、以下を提供します。
        - **Historical** フィールドでチャートする履歴データの日数。
        - チャートが終了するしきい値。

[ Set Options ] ウィンドウの完了後、[ **Next** ] を選択します。

1. Set Permissions ウィンドウを使用して、チャートの表示オプションを指定します。

- 許可されたすべてのユーザーがチャートを利用できるようにするには、**Share with all slider** を **Yes** に設定します。 **Accessmissions** フィールドで指定されたユーザーまたはグループへのアクセスを制限するには、**No** に設定します。 - **Access attempts** フィールドを使用して、チャートへのアクセスを付与するグループを選択します。

チャートの定義が終了したら、 **Finish** を選択して編集を保存し、**Manage Charts** タブのリストにチャートを追加します。

### 8.3.9 Dashboards

PEM は、多数のダッシュボードを介してパフォーマンス統計を表示します。各ダッシュボードには、選択したオブジェクトに関連する統計を表示するチャート、グラフ、表を含む一連の概要ビューが含まれています。

#### 8.3.9.1 ダッシュボードの概要

PEM クライアントは、PEM サーバーに接続すると、Global Overview ダッシュボードを表示します。追加のダッシュボードは、監視対象オブジェクトに関する統計情報を提供します。

##### ダッシュボードを開く

ダッシュボードは、PEM クライアントツリーコントロールに相当する階層で表示されます。ツリーコントロール内の各オブジェクトのダッシュボードには、そのオブジェクトと、ツリーコントロールのそのレベルの下にある監視対象オブジェクトの情報が必要に応じて表示されます。

各ダッシュボードヘッダーには、サーバーが起動された日時、関連する場合、ダッシュボードが最後に更新された日時、およびトリガーされたアラートの現在の数が表示されます。ダッシュボードヘッダーに表示されるナビゲーションメニューから、他のダッシュボードに簡単にアクセスできます。メニューは階層的に編成されています。ツリーコントロールで現在強調表示されているオブジェクトに適切なメニューのみが使用できます。

- ダッシュボードから **Global Overview** を選択して、Global Overview ダッシュボードに戻ります。
- **Agents** メニューからエージェントの名前を選択して、そのエージェントのオペレーティングシステム分析ダッシュボードに移動します。
- **Servers** メニューからサーバー名を選択して、そのサーバーの Server Analysis ダッシュボードに移動します。
- **Databases** メニューからデータベース名を選択して、そのデータベースのデータベース分析ダッシュボードに移動します。
- **Dashboards** メニューを使用して、グローバルレベルまたは選択したエージェント、サーバー、またはデータベースの情報ダッシュボードに移動します。

ダッシュボードには、次の形式で統計情報が表示されます。

- PEM プローブによって収集された統計情報を提供するテーブル。
- プローブの最新の実行によって収集された情報を表示する円グラフ。
- プローブの最新の実行によって収集された比較統計を表示する棒グラフ。
- PEM プローブによって収集された統計データを表示する折れ線グラフ。

次のいずれかの手法を使用して、ダッシュボードを開くことができます。

- **Management Dashboards** メニューから、アクティブなダッシュボード名を選択します。

- ツリー内の監視対象オブジェクトの名前を右クリックします。**Dashboards** メニューから、確認するダッシュボードの名前を選択します。

各ダッシュボードは、クライアントウィンドウのメインパネルの **Monitoring** タブに表示されます。ダッシュボードを開いた後、同じタブ内の他のダッシュボードに移動できます。

各ダッシュボードヘッダーには、他のダッシュボードに移動できるナビゲーションメニューが含まれています。ブラウザーの [進む] および [戻る] ボタンを使用して、以前に表示したダッシュボードをスクロールします。**Refresh** を使用して、現在のダッシュボードを更新します。

表形式で提供される統計を並べ替えるには、列見出しを選択します。並べ替え順序を逆にするには、それを再度選択します。各テーブルは、安定したソート機能を提供します。たとえば、各ユーザー名グループのセッション ID の昇順でテーブルをソートするには、最初に Session ID 列でソートし、次に User Name 列でソートします。

各グラフ、チャート、またはテーブルの右上隅にカーソルを合わせると、PEM クライアントツールバーアイコンが表示されます。アイコンにカーソルを合わせると、アイコンの機能を簡単に説明するツールチップが表示されます。

- **Refresh** アイコンを使用して、ダッシュボードに表示される情報を更新します。
- **Save Chart as Image** アイコンを使用して、選択したチャートを `.jpeg` 画像として保存します。
- **フルスクリーン**アイコンを使用してグラフを拡大し、グラフデータの詳細を表示します。
- **Personalize the chart configuration** アイコンを使用して、チャート固有の表示の詳細を選択できるコントロールパネルにアクセスします。
- **Explain** アイコンにカーソルを合わせて、グラフまたはチャートで共有されている情報の説明を確認します。

各グラフまたはチャートの右下隅には、グラフまたはチャートにプロットされた各項目を識別する凡例があります。

表示されている場合は、チャートの左上隅にある情報アイコンを選択して、チャートのコンテンツに関するメモと、必要に応じて、チャートのコンテンツを取得する 1 つ以上のプローブを有効にできるリンクを表示します。

#### 8.3.9.2 使用可能なダッシュボード

PEM は次のダッシュボードを提供しています。

##### アラートダッシュボード

アラートダッシュボードには、現在トリガーされているアラートが表示されます。グローバル概要から開いた場合、ダッシュボードには、システム上のすべての監視対象ノードの現在のアラートが表示されます。サーバー内のノードから開いた場合、レポートには、そのノードと、ツリー内のそのオブジェクトの下にあるすべての監視対象オブジェクトに関連するアラートが表示されます。

#### 監査ログ分析ダッシュボード

EDB Postgres Advanced Server ユーザーの場合、 Audit Log Analysis ダッシュボードを使用して、監査ログと収集が有効になっているインスタンスから収集された監査ログを参照できます。

#### PGD 管理ダッシュボード

PGD Admin ダッシュボードには、EDB Postgres 分散ノード、グループ、およびワーカーに関する概要情報が表示されます。

#### PGD グループモニタリングダッシュボード

PGD Group Monitoring ダッシュボードには、EDB Postgres 分散グループサブスクリプションとグループレプリケーションスロットに関する情報が表示されます。

#### PGD ノードモニタリングダッシュボード

PGD ノードモニタリングダッシュボードには、EDB Postgres 分散ノードスロット、ノードレプリケーションレート、および選択したノードの競合履歴の概要に関する情報が表示されます。

#### データベース分析ダッシュボード

Database Analysis ダッシュボードには、選択したデータベースのパフォーマンス統計が表示されます。

#### I/O 分析ダッシュボード

I/O 分析ダッシュボードには、オブジェクト DML アクティビティやログ操作など、さまざまな領域の I/O アクティビティが表示されます。

#### メモリ分析ダッシュボード

Memory Analysis ダッシュボードは、Postgres サーバーのさまざまなメモリ関連のメトリックに関する統計を提供します。

#### オブジェクトアクティビティ分析ダッシュボード

オブジェクトアクティビティ分析ダッシュボードは、選択したデータベースのテーブル/インデックスのパフォーマンスの詳細を提供します。

#### オペレーティングシステム分析ダッシュボード

オペレーティングシステム分析ダッシュボードは、基になるマシンのオペレーティングシステムのパフォーマンスに関する情報を提供します。

#### プローブログ分析ダッシュボード

Probe Log Analysis ダッシュボードには、PEM エージェントから返されたエラーメッセージが表示されます。

#### サーバー分析ダッシュボード

Server Analysis ダッシュボードは、選択した Postgres サーバーの全体的な操作に関する一般的なパフォーマンス情報を提供します。

#### サーバーログ分析ダッシュボード

Server Log Analysis ダッシュボードを使用すると、PEM サーバーに保存されているサーバーログの内容をフィルタリングおよび確認できます。

#### セッションアクティビティ分析ダッシュボード

セッションアクティビティ分析ダッシュボードは、選択したサーバーのセッションワークロードとロックアクティビティに関する情報を提供します

#### セッション待機分析ダッシュボード

Session Waits Analysis ダッシュボードは、 Advanced Server セッションの現在の DRITA 待機イベントの概要を提供します。

#### ストレージ分析ダッシュボード

Storage Analysis ダッシュボードには、テーブルスペースとオブジェクトのスペース関連のメトリックが表示されます。

#### システム待機分析ダッシュボード

System Waits Analysis ダッシュボードには、EDB Postgres Advanced Server セッションのシステム待機情報のグラフィカル分析が表示されます。

#### ストリーミングレプリケーション分析ダッシュボード

Streaming Replication Analysis ダッシュボードには、監視対象サーバーの WAL アクティビティに関する統計情報が表示され、フェールオーバーマネージャークラスターのステータスを監視できます。

### 8.3.9.3 ダッシュボード構成

[ Dashboard Configuration] ダイアログボックスのオプションを使用すると、ダッシュボード上のすべての折れ線グラフのタイムラインをリンクできます。[ Dashboard Configuration] ダイアログボックスを開くには、ダッシュボードヘッダーの **Settings** を選択します。

[ Dashboard Configuratio] ダイアログボックスを使用して、ダッシュボードに表示されるチャートの属性を制御します。

- 指定されたタイムラインをダッシュボードに表示される折れ線グラフに適用するには、**すべての折れ線グラフのタイムラインを** 有効にする **を設定します。**Disable**に設定されている場合、設定は後で使用できるように保存されますが、表示されるデータの量は変更されません。
- **Days** セレクターを使用して、折れ線グラフに表示する収集データの日数を指定します。
- **Hour(s)** セレクターを使用して、折れ線グラフに表示する収集データの時間数を指定します。

- **このダッシュボードの構成を記憶する**を選択して、カスタマイズされた時間スパンを現在のダッシュボードにのみ適用します。すべてのダッシュボードの折れ線グラフにタイムスパンをグローバルに適用することはオフのままにします。

[ Dashboard Configuration] ダイアログボックスで指定した設定は、現在のユーザーのセッションにのみ適用されます。

設定を指定したら、[ **Save** ] を選択します。

#### 8.3.9.4 カスタムダッシュボード

PEM は、多数のシステム定義のダッシュボードにパフォーマンス統計を表示します。各ダッシュボードには、選択したオブジェクトに関連する統計を表示するチャート、グラフ、表を含む一連の概要ビューが含まれています。選択したグラフとレイアウトを使用してカスタムダッシュボードを作成することもできます。

カスタムダッシュボードを作成するには、最初に作成するダッシュボードと同じレベルで既存のダッシュボードを開きます。たとえば、新しいサーバーレベルのダッシュボードを作成する場合は、既存のサーバーレベルのダッシュボードのいずれかを開きます。

次に、ダッシュボードの右上にある . . . メニューをクリックし、**Create Dashboard** を選択します。これにより、ダッシュボードをデザインできるビジュアルエディタが開きます。

完了したら、[ **Save** ] をクリックします。ダッシュボードは、画面の上部のパンくずリストバーに表示されるダッシュボードのリストに表示されます。

既存のダッシュボードを変更するには、 **Edit** を選択してビジュアルエディターを再度開きます。

ダッシュボードを削除するには、ダッシュボード名の横にある **Delete** を選択します。

既存のダッシュボードの権限を変更するには、 **Share Dashboard** を選択します。

このダッシュボードを表示できる人を選択するには、ダッシュボード名の横にある歯車アイコンをクリックします。デフォルトでは、ダッシュボードはすべてのユーザーと共有されます。ダッシュボードをプライベートにしたり、特定のロールに制限したりできます。

### 8.3.10 Remote monitoring

リモートモニタリングは、別のホストにある PEM エージェントを使用して Postgres クラスターを監視することです。

PEM で Postgres クラスターをリモートでモニタするには、クラスターを PEM に登録し、PEM エージェントをバインドする必要があります。詳細については、*Registering a Postgres server* を参照してください。

次のシナリオでは、PEM を使用したリモート監視が必要です。

- AWS RDS で実行されている Postgres クラスター
- Postgres cluster running on BigAnimal

PEM リモート監視は以下をサポートしています。

| Feature Name | Remote monitoring supported? | Comments |
|---|---|---|
| *Platform Compatibility* | Yes | NA |
| *Viewing system reports* | Yes | NA |
| *Platform Compatibility* | Limited | Postgres クラスターとオペレーティングシステムのメトリックの間には、相関関係がありません。 |
| *Platform Compatibility* | Limited | Postges クラスターでアラートスクリプトを実行すると、実際の Postgres クラスターマシンではなく、バインドされた PEM エージェントが実行されているマシンで実行されます。 |
| *Platform Compatibility* | Limited | 一部のダッシュボードは、完全なデータを表示できない場合があります。たとえば、Postgres クラスターが実行されているオペレーティングシステム情報は利用できないため、表示されません。 |
| Manage probes | Limited | 一部の PEM プローブは情報を結果ず、一部の機能が影響を受ける可能性があります。プローブ機能の詳細については、*Platform Compatibility* を参照してください。 |
| *Platform Compatibility* | Limited | スケジュールされたタスクは Postgres クラスターでのみ動作し、スクリプトはリモートエージェントで実行されます。 |
| *Platform Compatibility* | Limited | コア使用状況レポートには、完全な情報が表示されません。たとえば、プラットフォーム、コア数、および合計 RAM は表示されません。 |
| *Platform Compatibility* | No | NA |
| *Platform Compatibility* | No | NA |

## 8.4 Monitoring Barman

Barman Backup and Recovery Manager は、ビジネスクリティカルな環境での PostgreSQL サーバーのリモートバックアップとディザスターリカバリーのためのオープンソース管理ツールです。PostgreSQL のポイント インタイムリカバリーテクノロジーに依存しているため、DBA はバックアップの完全なカタログと、複数のリモートサーバーのリカバリフェーズを 1 つの場所からリモートで管理できます。詳細については、*Monitoring Barman* を参照してください。

PEM コンソールを介して Barman サーバーをモニタできます。

### 8.4.1 Barman をモニタリングするための前提条件

Barman サーバーを PEM コンソールに追加する前に

- Barman ホストに Barman を手動でインストールして構成する必要があります。Barman のインストールと構成の詳細については、*Monitoring Barman* を参照してください。
- Barman ホストに pg-backup-api ツールをインストールします。インストールの詳細については、pg-backup-api を参照してください。

### 8.4.2 Barman サーバーの構成

次を使用して Barman サーバーを構成および編集できます。

- PEM Web クライアント
- `pemworker` コマンドライン

#### 8.4.2.1 PEM Web クライアントを使用する

#### 8.4.2.2 構成

[ Create > Barman Server ] ダイアログボックスを使用して、既存の Barman サーバーを PEM サーバーに登録できます。ダイアログボックスを開くには、Barman Servers ノードを右クリックし、 **Create > Barman Server** を選択します。

**General** タブを使用して、Barman サーバーの一般的なプロパティを説明します。

- **Name** フィールドを使用して、サーバーの名前を指定します。名前は、ブラウザツリー内のサーバーを識別します。
- **URL** フィールドを使用して、Barman がインストールされているホストの URL を指定します。
- **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人、または PEM サーバーのスーパーユーザー権限を持つ PEM ユーザーのみが、PEM にログオンするとこのサーバーを表示できます。このフィールドを空白のままにした場合、すべての PEM ユーザーにサーバーが表示されます。

[ **PEM Agent** ] 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。

- **Bound Agent** フィールドを使用して、Barman サーバーとして構成するエージェントを選択します。Barman でサポートされている PEM エージェントのみがリストされています。
- **Probe Frequency** フィールドを使用して、指定した間隔でプローブを実行する秒数を指定します。
- **Heartbeat** フィールドを使用して、PEM エージェントの可用性を秒単位で確認する間隔を指定します。

> **注釈**
>
> Barman サーバーを登録した後、PEM エージェントを再起動する必要があります。

#### 8.4.2.3 編集

Barman サーバーを編集するには、ブラウザツリーからサーバーを右クリックし、**プロパティ** を選択します。

- **PEM Agent** タブを使用して、バインドされたエージェント、プローブ周波数、およびハートビートを変更します。Barman サーバーの所有者のみが [ **PEM エージェント**] タブのフィールドを変更できます。
- **Information** タブを使用して、Barman サーバーに関する詳細情報を表示します。このタブには、Barman 関連のプローブが実行されるたびに値が設定されます。
- **Configuration** タブを使用して、Barman サーバーの構成設定を表示します。このタブには、Barman 関連のプローブが実行されるたびに値が設定されます。

#### 8.4.2.4 pemworker コマンドラインを使用する

pemworker コマンドラインオプションを使用して Barman サーバーを構成できます。

```
$ ./pemworker --update-barman --help
./pemworker --update-barman [barman-update-options]

barman-update-options:
  --id <barman-id> (ID for the existing Barman API pg-backup-api)
  --api-url <url> (URL of the Barman API pg-backup-api)
  --probe-execution-frequency <interval> (Default: 30, Probe the Barman API pg-backup-
↪api at regular interval in seconds and fetch the metrics.)
  --heartbeat-interval <interval> (Default: 10, Ping the Barman API pg-backup-api␣
↪status API at a regular interval in seconds for checking its availability.)
  --ssl-crt <certificate_file> (SSL certificate file for the Barman API.)
  --ssl-key <key_file> (Private SSL key for the Barman API.)
  --ssl-ca-cert <ca_file> (CA certificate to verify peer against the Barman API.)
  --config-file/-c <config_file> (Path to the agent configuration file.)

$ ./pemworker --unregister-barman --help
./pemworker --unregister-barman [barman-unregistration-options]

barman-unregistration-options:
  --id <barman-id> (ID for the existing Barman API, registered with the PEM Server.pg-
↪backup-api)
  --config-file/-c <config_file> (Path to the agent configuration file.)

$ ./pemworker --register-barman --help
./pemworker --register-barman [barman-registration-options]

barman-registration-options:
  --api-url <url> (URL of the Barman API pg-backup-api)
  --description <name> (Description to show on the UI User interface for the Barman␣
↪API.)
  --probe-execution-frequency <interval> (Default: 30, Probe the Barman API pg-backup-
↪api at regular interval in seconds and fetch the metrics.)
  --heartbeat-interval <interval> (Default: 10, Ping the Barman API pg-backup-api␣
↪status API at a regular interval in seconds for checking its availability.)
  --ssl-crt <certificate_file> (SSL certificate file for the Barman API.)
  --ssl-key <key_file> (Private SSL key for the Barman API.)
  --ssl-ca-cert <ca_file> (CA certificate to verify peer against the Barman API.)
  --team <database-role> (Specify the name of the database group role, on the PEM␣
↪backend database server, that should have access to this Barman API Server.)
  --owner <database-user> (Specify the name of the database user, on the PEM backend␣
↪database server, who will own the Barman API Server.)
  --config-file/-c <config_file> (Path to the agent configuration file.)
```

> **注釈**
>
> Barman サーバーを登録した後、PEM エージェントを再起動する必要があります。

### 8.4.3 PEM ダッシュボードでの Barman サーバーの詳細の表示

Barman サーバーが構成されると、その Barman サーバーのバックアップ全体とサーバー関連の詳細を PEM ダッシュボードで表示できます。

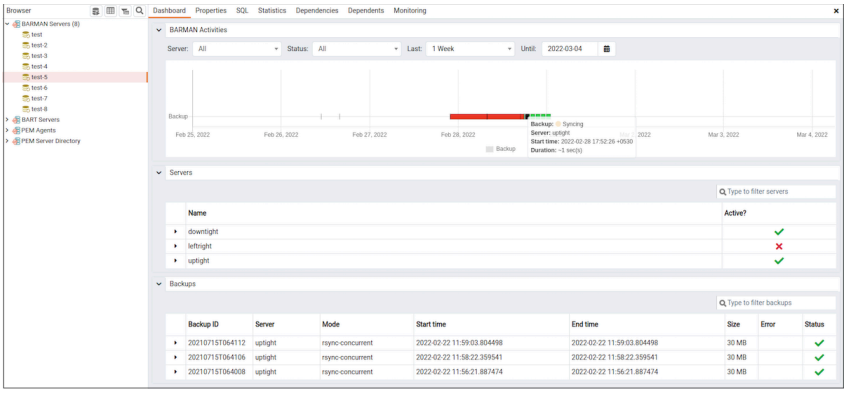



図 12: Barman dashboard

監視対象の Barman サーバーを選択すると、関連するすべてのデータベースサーバーの詳細とそのアクティビティが Barman アクティビティパネルのダッシュボードにグラフとして表示されます。フィルタボックスで指定した任意の基準データベースサーバー、ステータス、期間、または日付に基づいてアクティビティを選択できます。

[ Servers ] パネルには、その Barman サーバーによって管理されているすべてのデータベースサーバーのリストがアクティブなステータスとともに表示されます。

[ Backups ] パネルには、その Barman サーバーによって管理されているすべてのデータベースサーバーのバックアップのリストが表示されます。リストをフィルタリングして、データベースサーバーの詳細を表示できます。フィルタボックスで指定した基準でリストをフィルタリングすることもできます。通常、このフィルタは、列の下にリストされている任意の種類の文字列値日付、時刻、サイズを除きます。たとえば、`tar` と入力してリストをフィルタリングし、tar 形式のバックアップのみを表示できます。

バックアップの詳細には、バックアップ ID、サーバー、モード、開始時刻、終了時刻、サイズ、エラー、ステータス列が含まれます。

## 8.5 Monitoring EDB Postgres Distributed

EDB Postgres 分散は、高度な競合管理、データ損失保護、および throughput up to 5X faster than native logical replication を備えたマルチマスターレプリケーションとデータ分散を提供し、最大ファイブナインの高可用性を備えた分散 PostgreSQL クラスターを有効にします。PEM コンソールを介して EDB Postgres 分散クラスター内のノードを監視する前に、最初に EDB Postgres 分散クラスターを展開し、データベースノードが起動して実行されていることを確認する必要があります。EDB Postgres 分散のインストールの詳細については、*Monitoring EDB Postgres Distributed* を参照してください。

PEM バージョン 8.1.0 および EDB Postgres Distributed バージョン 3.7.9 以降のダッシュボードを使用して、1 つ以上の EDB Postgres 分散データベースノードに関するステータス情報を表示するように PEM を構成できます。

EDB Postgres 分散データベースノードを監視するように PEM を構成するには、PEM Web クライアントを使用してサーバー定義を作成します。**データベースサーバーの手動登録** ダイアログボックスのタブを使用して、EDB Postgres 分散データベースノードの一般的な接続プロパティを指定します。ただし、次の例外。

- [ **PEM Agent** ] タブの **Database** フィールドに EDB Postgres Distributed 対応データベース名を指定します。
- [ **PEM Agent** ] タブの **username** フィールドに pgd_monitor または pgd_superuser ロールを持つユーザーを指定します。

サーバー定義を保存すると、EDB Postgres 分散データベースノードは、PEM クライアントオブジェクトブラウザツリーの PEM サーバーディレクトリの下のサーバーのリストに含まれます。EDB Postgres DistributedPGD ダッシュボードからノードをモニタできます。

EDB Postgres DistributedPGD ダッシュボードにモニタリング情報を含めるには、各 EDB Postgres Distributed グループの相対プローブを有効にする必要があります。EDB Postgres 分散 **プローブ** の完全なリストを参照してください。

プローブを有効にするには、ノード名を右クリックし、 **Management > Manage Probes** を選択します。

**注釈**

*PGD ノードレプリケーションレート*プローブは、EDB Postgres Advanced Server および EDB Postgres ExtendedPGE でのみ使用可能な機能である PGD ラグトラッキングに依存しています。このプローブは、コミュニティ PostgreSQL ではサポートされていません。コミュニティ PostgreSQL の PGD クラスターに対して実行すると、*apply_rate* および `catchup_interval` メトリックは使用できず、エージェントログは `lag tracking is not supported with this variant of Postgres` をレポートします。この制限は、メトリックレポートにのみ影響します。レプリケーションは通常に続行されます。基本的なレプリケーションラグメトリック `replay_lag` 、*replay_lag_bytes* は、すべての PostgreSQL バリアントで*PGD ノードスロット*プローブを介して利用できます。

EDB Postgres 分散データベースノードをモニタするには、オブジェクトブラウザツリーでノードの名前を右クリックします。**Dashboards** メニューから、PGD Admin、PGD Group Monitoring、または PGD Node Monitoring ダッシュボードを選択します。

## 8.6 Cluster topology

クラスタートポロジは、PostgreSQL レプリケーションクラスターを視覚的に表現します。ストリーミングレプリケーションを介して接続されているデータベースサーバーを自動的に検出し、以下を示す対話型図に表示します。

- プライマリサーバー — レプリケーションのソース。
- レプリカサーバー — レプリケートされたデータを受信するサーバー。
- レプリケーション関係 — どのサーバーがどのサーバーから複製するか。

- 同期状態 — レプリケーションが同期か非同期か。
- カスケードレプリケーション — 他のレプリカから複製するレプリカ。
- PGD / BDR クラスター —すべてのノードが他のすべてのノードにレプリケートするメッシュレプリケーション。
- Witness ノード —高可用性 HA セットアップでコンセンサスに使用される非データノード。

### 8.6.1 トポロジ検出の仕組み

PEM は、PostgreSQL サーバーからデータを収集する専用プローブを使用して、レプリケーショントポロジを検出します。プローブは、生のレプリケーションデータを収集し、相関機能によって処理されて、クラスタートポロジを構築します。

> **注釈**
>
> **Refresh**をクリックすると、トポロジビューが更新されます。これにより、相関ファンクションがトリガーされ、最新のプローブデータを処理し、クラスター情報を更新します。

#### 8.6.1.1 必須プローブ

トポロジ機能が正常に動作するには、監視対象サーバーで次のプローブを有効にします。

| Probe name | Required for | Description |
|---|---|---|
| Streaming Replication | All cluster types | `pg_stat_replication`プライマリ上および`pg_stat_wal_receiver`スタンバイ上からレプリケーションデータを収集します。すべてのトポロジ検出に必要です。 |
| PGD Group Summary (*bdr_group_summary*) | PGD clusters | PGD グループレベルの情報を収集します。 |
| PGD Node Summary (*bdr_node_summary*) | PGD clusters | ノード名、サブグループ、書き込みリーダーステータスを含む PGD ノード情報を収集します。 |
| EFM Cluster Node Status | EFM clusters | `efm_cluster_node_status`プローブを介して EFM ノードデータを収集します。 |

これらのプローブは、新しいサーバー登録でデフォルトで有効になっています。トポロジデータが表示されない場合は、これらのプローブが **Management > Probe Configuration** で有効になっていることを確認します。

#### 8.6.1.2 クラスタータイプの検出

PEM は、サーバー構成に基づいてクラスタータイプを自動的に検出します。

| Cluster type | How it's detected |
|---|---|
| Physical Streaming Replication (PSR) | Default for servers with streaming replication and no HA manager configured. |
| PGD | Servers with the BDR extension installed (detected via PGD probes). |
| Patroni | Servers with **Patroni Cluster Name** configured in server properties. |
| EDB Failover Manager (EFM) | Servers with **EFM Cluster Name** configured in server properties. |

Patroni または EFM クラスター検出を有効にするには、PEM でサーバーを登録または編集するときにサーバーのプロパティで各クラスター名を構成します。

### 8.6.2 トポロジービューへのアクセス

2 つの方法でクラスタートポロジビューを開くことができます。

- パネルヘッダーの右上隅にある **3 つの点⊠** をクリックし、 **Open > Cluster Topology** を選択します。
- オブジェクトツリーでサーバーを右クリックし、 **Show Cluster Topology** を選択します。

#### 8.6.2.1 必要な権限

クラスタートポロジを表示するには、ユーザーに `pem_topology_view` ロールが必要です。 PEM 管理者に連絡して、この権限を付与してください。

### 8.6.3 トポロジ表示を理解する

#### 8.6.3.1 クラスターリスト左側のサイドバー

サイドバーには、検出されたすべてのレプリケーションクラスターが表示されます。

| Element | Description |
|---|---|
| Cluster name | 自動生成またはプライマリサーバーの説明。 |
| Type badge | PSR、パトローニ、EFM、または PGD。 |
| Member count | クラスター内のサーバーの数。 |

#### 8.6.3.2 サポートされているクラスタータイプ

| Type | Description |
|---|---|
| PSR | 物理的ストリーミングレプリケーション標準の PostgreSQL。 |
| Patroni | Patroni 管理対象 HA クラスター。 |
| EFM | EDB フェールオーバーマネージャークラスター。 |
| PGD | メッシュレプリケーションを使用した Postgres DistributedBDR。 |

#### 8.6.3.3 トポロジグラフメインパネル

インタラクティブ図には、ノードとそれらの間の接続が表示されます。

**ノードタイプ**

| Node color | Description |
|---|---|
| Green border | プライマリ — レプリケーションを提供するソースサーバー。 |
| Blue border | レプリカ — レプリケートされたデータを受信するサーバー。 |
| Purple border | データノード — 読み取りおよび書き込みを受け入れる PGD データノード。 |
| Orange border | Witness — HA コンセンサス用の非データノード。 |
| Gray border | Unknown — 未登録のレプリカ IP アドレスを表示します。 |

各ノードには、サーバーの名前または説明、ホスト名とポート、ロールバッジプライマリ/レプリカ、およびレプリカの同期状態バッジが表示されます。

図の右上隅の凡例は、色分けを説明しています。

### 8.6.4 インタラクティブ SVG ビュー

トポロジの視覚化では、次の機能を備えたインタラクティブなスケーラブルベクターグラフィック SVG ビューを使用します。

#### 8.6.4.1 パンとズーム

- **ズームイン/アウト** — マウススクロールホイールを使用します。
- **パン** — キャンバス上の任意の場所をクリックしてドラッグします。
- **ズームコントロール** — 右下隅にある **+** および **−** ボタンを使用します。 **Fit to screen** ボタンは、ビューをリセットして、ビューポート内のすべてのノードに収まるようにします。

#### 8.6.4.2 フルスクリーンモード

ヘッダーの **フルスクリーン**ボタンをクリックして、ビューを展開します。これは、チームメンバーにトポロジを提示したり、大規模なクラスターを表示したり、ドキュメントのスクリーンショットを撮ったりする場合に役立ちます。**Escape** を押すか、ボタンを再度クリックして終了します。

#### 8.6.4.3 SVG エクスポート

ヘッダーの **Download** ボタンをクリックして、現在のトポロジビューを SVG ファイルとしてエクスポートします。

#### 8.6.4.4 ノードカードの詳細

各サーバーは、次の情報を含むカードとして表示されます。

- **アイコン** —登録済みサーバーのストレージアイコン、未登録サーバーのヘルプアイコン。

- **サーバー名** — サーバーの説明またはホスト名。
- **ロールバッジ** P プライマリ、R レプリカ、D PGD のデータノード、W 書き込みリーダーまたは Witness、? 不明。
- **Hostname:Port** — サーバーの接続情報。
- **タグ** — 「Write Leader」や「Raft_Leader」などのサーバータグは、丸薬型のバッジとして表示されます。
- **ステータス** — PEM に登録されていないサーバーの「Not Registered」インジケーター。

#### 8.6.4.5 レンダラー設定

PEM は 2 つのレンダリングモードを提供します。 **Preferences > Cluster Topology > Renderer** でそれらを切り替えます。

| Renderer | Description |
|---|---|
| SVG (default) | パン/ズームおよびスムーズなツールチップを備えたインタラクティブ SVG ビュー。 |
| HTML | パン/ズームを使用しないクラシックな HTML/CSS レンダリング。 |

### 8.6.5 PGD/ BDR クラスター

BDR 双方向レプリケーションとしても知られる Postgres DistributedPGD は、すべてのノードが書き込みを受け入れることができるマルチマスターレプリケーションソリューションです。従来のプライマリレプリカ設定とは異なり、PGD は、すべてのノードが他のすべてのノードにレプリケートするメッシュレプリケーションを使用します。

#### 8.6.5.1 ポリゴンとフォースグラフのレイアウト

**ポリゴンレイアウト 1〜8 ノード**

1〜8 のデータノードを含む PGD クラスターの場合、PEM はポリゴンメッシュレイアウトを使用します。1 つのノードには単一の中央カード、2 には水平ペア、3〜8 ノードにはポリゴン構成三角形、正方形、五角形など。破線はすべてのノードを接続して、メッシュレプリケーショントポロジを視覚化します。

**Force グラフレイアウト 9+ノード**

9 つ以上のデータノードを持つ大規模な PGD クラスターの場合、PEM は強制グラフレイアウトに切り替えます。ノードは、番号付きのラベルの付いた色付きの円として表示されます。ノードにカーソルを合わせると、ツールチップで完全な詳細が表示されます。書き込みリーダーは外側のリングで強調表示され、まばらなメッシュ線は、視覚的に混乱することなく接続を示します。

#### 8.6.5.2 サブグループ

PGD クラスターは、ノードをサブグループロケーションとも呼ばれ、通常は地理的なデータセンターまたはアベイラビリティーゾーンを表すサブグループに編成できます。PGD クラスターに複数のサブグループがある場合、PEM はそれらを、サブグループの中心を接続するサブグループ間破線で個別の境界線付きカードとして表示します。3 つ以上のサブグループの場合、カードは円形に配置されます。

すべてのノードが同じサブグループに属しているか、サブグループが定義されていない場合、PEM は標準のポリゴンまたはフォースグラフレイアウトを使用してノードを直接表示します。

### 8.6.6 未登録のレプリカでの作業

PEM が PEM 監視対象サーバーとして登録されていないレプリカを検出すると、カード内の IP アドレスと「Not Registered」ラベルを示す「不明な」ノードとして表示されます。

これを解決するには、PEM にレプリカサーバーを登録します。

### 8.6.7 トポロジデータの更新

Streaming Replication プローブは、デフォルトで 5 分ごとに実行され、サーバーから最新のレプリケーションデータを収集します。この生データは、相関機能で処理して、トポロジビューを更新する必要があります。

トポロジビューヘッダーの **Refresh** をクリックして相関機能をトリガーし、新しいまたは変更されたクラスターでクラスターリストを更新し、現在のトポロジグラフを更新します。レプリケーション設定に変更を加えた後-レプリカの追加やフェールオーバーの実行後、常に [ **Refresh** ] をクリックして、更新されたトポロジを表示します。

### 8.6.8 障害対応

**クラスターは表示されません**

考えられる原因

- Streaming Replication プローブはまだ実行されていません - 5 分間待機するか、手動更新をトリガーします。
- 監視対象サーバーではストリーミングレプリケーションが構成されていません。
- `streaming_replication` プローブは無効になっています。

プローブが **プローブ構成**で有効になっていることを確認し、`pg_stat_replication` がプライマリサーバー上のデータを返すことを確認して、更新を手動でトリガーします。

**サーバーは「Unknown」と表示されます**

レプリカの IP アドレスは、登録されたサーバーと一致しません。PEM にサーバーを登録するか、サーバーの `server` または `hostaddr` フィールドがレプリケーションで表示される IP と一致することを確認します。

**トポロジグラフは空です**

考えられる原因

- 選択したクラスターには、アクティブなメンバーがありません。

- すべてのメンバーが非アクティブになりました。

サーバーが実行され、エージェントが接続されていることを確認し、プローブがデータを収集していることを確認します。

```
SELECT * FROM pemdata.streaming_replication;
```

**PGD クラスターが検出されませんでした**

PGD Node Summary bdr_node_summary プローブはデータを収集していません。PGD サーバーでプローブが有効になっていることを確認し、サーバーに BDR がインストールおよび構成されていることを確認し、プローブデータを照会します。

```
SELECT * FROM pemdata.bdr_node_summary WHERE server_id = <id>;
```

**PGD ノードが「Unregistered」と表示される**

BDR ノード名は、登録されたサーバーの説明またはホスト名と一致しません。PGD Node Summary bdr_node_summary プローブでノード名を確認します。

```
SELECT node_name FROM pemdata.bdr_node_summary;
```

次に、ノード名と正確に一致するようにサーバーの説明を更新するか、正しいノード名を説明として新しいサーバーを登録します。

### 8.6.9 ベストプラクティス

- **すべてのサーバーを登録する**最も正確なトポロジビューを得るには、レプリカを含むすべてのサーバーを登録します。未登録のノードには、ヘルプアイコンと「未登録」ステータスが表示されます。
- **一貫した名前付けを使用します** - サーバーに意味のある説明を提供して、トポロジ図を理解しやすいようにします。
- **PGD クラスターの場合** — 適切なノード照合のために、BDR ノード名と正確に一致するようにサーバーの説明を設定します。
- **プローブのステータスをモニタする** — streaming_replication プローブが有効になっていて実行されていることを確認します。
- **フェールオーバー後のチェック** —計画的または計画外のフェールオーバーの後、トポロジを更新して新しい構成を確認します。
- **ドキュメントには SVG エクスポートを使用します** —アーキテクチャドキュメントまたはプレゼンテーションに含めるために、トポロジを SVG としてエクスポートします。
- **大規模なクラスターにはフルスクリーンを使用します** —多くのノードを持つクラスターの場合、フルスクリーンモードはより適切な概要を提供します。
- **詳細をズーム** —ズームコントロールを使用して、個々のノードの詳細を確認するか、クラスター全体の鳥瞰図を取得します。
- **ノードツールチップの確認**強制グラフレイアウトでは、ノードにカーソルを合わせると、ホスト名、ポート、ロール、タグを含む詳細情報を表示します。

### 8.6.10 トポロジデータの SQL クエリー

スクリプト、レポート、または他のツールとの統合のために、SQL を介してトポロジデータを直接照会する場合は、PEM データベースに対して次のクエリを実行します。

#### 8.6.10.1 検出されたすべてのクラスターをリストする

```
SELECT
    c.id AS cluster_id,
    c.name AS cluster_name,
    c.cluster_type,
    ps.description AS primary_server,
    ps.server AS primary_host,
    (SELECT COUNT(*) FROM pem.replication_cluster_member m
     WHERE m.cluster_id = c.id AND m.is_active = true) AS active_members,
    c.updated_at AS last_updated
FROM pem.replication_cluster c
LEFT JOIN pem.server ps ON ps.id = c.primary_server_id
ORDER BY c.name;
```

#### 8.6.10.2 ロールを持つクラスターメンバーを表示する

```
SELECT
    c.name AS cluster_name,
    c.cluster_type,
    m.role,
    COALESCE(s.description, m.client_hostname, host(m.client_addr)::text) AS node_
↪name,
    s.server AS hostname,
    s.port,
    m.sync_state,
    m.cascade_level,
    m.subgroup_name,
    CASE WHEN m.server_id IS NOT NULL THEN Yes ELSE No END AS registered_in_pem,
    m.updated_at
FROM pem.replication_cluster_member m
JOIN pem.replication_cluster c ON c.id = m.cluster_id
LEFT JOIN pem.server s ON s.id = m.server_id
WHERE m.is_active = true
ORDER BY c.name, m.cascade_level, m.role DESC;
```

#### 8.6.10.3 レプリケーションラグの表示 PSR、EFM、および Patroni クラスター

```
SELECT
    c.name AS cluster_name,
    ps.description AS primary_server,
    sr.client_addr AS replica_address,
    COALESCE(rs.description, sr.client_addr) AS replica_name,
    m2.sync_state,
    sr.lag_mb,
    sr.recorded_time AS last_collected
FROM pemdata.streaming_replication sr
JOIN pem.replication_cluster_member m
    ON m.server_id = sr.server_id AND m.role = primary AND m.is_active = true
JOIN pem.replication_cluster c ON c.id = m.cluster_id
JOIN pem.server ps ON ps.id = sr.server_id
LEFT JOIN pem.server rs ON host(rs.server::inet) = host(sr.client_addr::inet)
LEFT JOIN pem.replication_cluster_member m2
    ON m2.cluster_id = c.id AND host(m2.client_addr) = host(sr.client_addr::inet)
    AND m2.is_active = true
WHERE c.cluster_type IN (psr, efm, patroni)
  AND sr.client_addr IS NOT NULL
ORDER BY c.name, sr.lag_mb DESC;
```

#### 8.6.10.4 レプリケーションラグ PGD クラスターの表示

```
SELECT
    c.name AS cluster_name,
    ps.description AS write_leader,
    bns.target_name AS peer_node,
    bns.client_addr AS peer_address,
    bns.active_pid AS slot_pid,
    bns.replay_lag_bytes,
    ROUND(bns.replay_lag_bytes / (1024.0 * 1024), 2) AS lag_mb,
    bns.replay_lag_size AS lag_pretty,
    bns.recorded_time AS last_collected
FROM pemdata.bdr_node_slots bns
JOIN pem.replication_cluster_member m
    ON m.server_id = bns.server_id AND m.role = primary AND m.is_active = true
JOIN pem.replication_cluster c ON c.id = m.cluster_id
JOIN pem.server ps ON ps.id = bns.server_id
WHERE c.cluster_type = pgd
  AND bns.target_name IS NOT NULL
ORDER BY c.name, lag_mb DESC;
```

#### 8.6.10.5 完全なトポロジの概要すべてのクラスタータイプ

このクエリは、すべてのクラスター、メンバー、およびレプリケーションラグの統合ビューを提供します。

```
WITH cluster_lag AS (
    -- PSR/EFM/Patroni lag from streaming_replication
    SELECT
        m.cluster_id,
        host(sr.client_addr::inet) AS replica_addr,
        sr.lag_mb
    FROM pemdata.streaming_replication sr
    JOIN pem.replication_cluster_member m
        ON m.server_id = sr.server_id AND m.role = primary AND m.is_active = true
    JOIN pem.replication_cluster c ON c.id = m.cluster_id
    WHERE c.cluster_type IN (psr, efm, patroni)
      AND sr.client_addr IS NOT NULL

    UNION ALL

    -- PGD lag from bdr_node_slots
    SELECT
        m.cluster_id,
        host(bns.client_addr::inet) AS replica_addr,
        ROUND(bns.replay_lag_bytes / (1024.0 * 1024), 2) AS lag_mb
    FROM pemdata.bdr_node_slots bns
    JOIN pem.replication_cluster_member m
        ON m.server_id = bns.server_id AND m.role = primary AND m.is_active = true
    JOIN pem.replication_cluster c ON c.id = m.cluster_id
    WHERE c.cluster_type = pgd
      AND bns.target_name IS NOT NULL
)
SELECT
    c.name AS cluster_name,
    c.cluster_type,
    m.role,
    COALESCE(s.description, m.client_hostname, host(m.client_addr)::text) AS node_
↪name,
    s.server AS hostname,
    s.port,
    m.sync_state,
    m.subgroup_name,
    COALESCE(cl.lag_mb, 0) AS lag_mb,
    CASE WHEN m.server_id IS NOT NULL THEN Yes ELSE No END AS registered
FROM pem.replication_cluster_member m
JOIN pem.replication_cluster c ON c.id = m.cluster_id
LEFT JOIN pem.server s ON s.id = m.server_id
LEFT JOIN cluster_lag cl
```

(次のページに続く)

(前のページからの続き)

```
    ON cl.cluster_id = m.cluster_id
    AND cl.replica_addr = COALESCE(host(m.client_addr), s.server)
WHERE m.is_active = true
ORDER BY c.name, m.cascade_level, m.role DESC;
```

## 8.7 Monitoring Failover Manager

EDB Failover Manager を使用してレプリケーションシナリオをモニタしている場合、Failover Manager を手動でインストールおよび構成する必要があります。 Failover Manager のインストールの詳細については、EDB website を参照してください。

ストリーミングレプリケーションダッシュボードでフェールオーバーマネージャークラスターのステータスを監視するには、クラスターの各ノードの **Server Properties** ダイアログボックスの **Advanced** タブで次の情報を提供する必要があります。

- **EFM Cluster Name** フィールドを使用して、Failover Manager クラスターの名前を指定します。クラスター名は、クラスタープロパティファイルの名前の接頭辞です。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。
- **EFM Installation Path** 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、Failover Manager バイナリファイルは`/usr/edb/efm-<X>/bin` にインストールされます。`<X>` は EFM バージョンです。

> **注釈**
>
> フェールオーバーマネージャーを監視するために、PEM エージェントはユーザー efm として`efm cluster-status-json <cluster_name>` コマンドを実行します。クラスタープロパティファイルが efm ユーザーのホームディレクトリにない場合、このコマンドは失敗します。この場合、efm のホームディレクトリにクラスタープロパティファイルへのシンボリックリンクを作成し、efm がクラスタープロパティファイルを読み取りおよび実行する権限があることを確認します。

サーバーを登録すると、Streaming Replication Analysis ダッシュボードのダッシュボードの下部近くに EFM クラスターに関するステータス情報が表示されます。

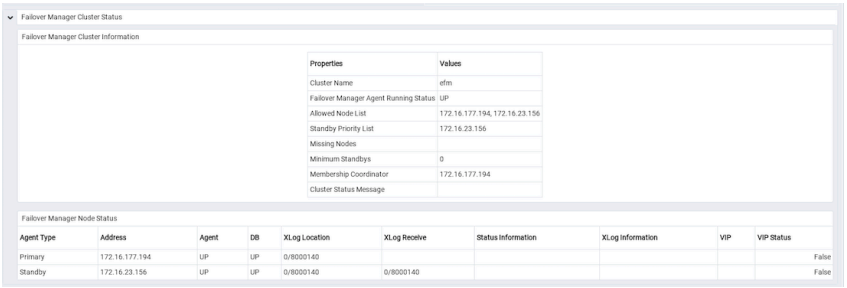



図 13: The Failover Manager cluster status report

Streaming Replication Analysis ダッシュボードの **Failover Manager Cluster Status** セクションには、監視対象クラスターに関する情報が表示されます。

Failover Manager Cluster Information テーブルは、Failover Manager クラスターに関する情報を提供します。

- Properties 列には、クラスタープロパティの名前が表示されます。
- Values 列には、プロパティの現在の値が表示されます。

[Failover Manager Node Status] テーブルには、Failover Manager クラスターの各ノードに関する情報が表示されます。

- Agent Type 列には、ノードにあるエージェントの種類が表示されます。使用可能な値は、プライマリ、レプリカ、Witness、Idle、および Promoting です。
- Address 列には、ノードの IP アドレスが表示されます。
- Agent 列には、ノードにあるエージェントのステータスが表示されます。
- DB 列には、ノードにあるデータベースの状態が表示されます。
- XLog Location 列には、データベースのトランザクションログの場所が表示されます。
- Status Information 列には、ノードに関するエラー関連の情報が表示されます。
- XLog Information 列には、トランザクションログに関するエラー関連の情報が表示されます。
- VIP 列には、ノードに関連付けられた VIP アドレスが表示されます。
- VIP Status 列には、VIP がノードでアクティブな場合は True が表示され、そうでない場合は False が表示されます。

### 8.7.1 プライマリノードの置き換え

PEM クライアントを使用して、フェールオーバーマネージャークラスターのプライマリノードをレプリカノードに置き換えることができます。フェールオーバープロセスを開始するには、**Tools > Server > Replace Cluster Primary** を選択します。現在のプライマリノードを置き換えることを確認するプロンプトが表示されます。

**Yes** を選択して、現在のプライマリノードをフェールオーバーマネージャークラスターから削除し、レプリカノードをフェールオーバーマネージャークラスターの読み取り/書き込みプライマリノードの役割に昇格させます。最も高いプロモーション優先度フェールオーバーマネージャーで定義されたノードが、新しいプライマリノードになります。PEM はジョブステータスを報告します。

ジョブが完了し、ストリーミングレプリケーション分析ダッシュボードが更新されたら、フェールオーバーマネージャーノードのステータステーブルを確認して、レプリカノードがフェールオーバーマネージャークラスター内のプライマリの役割に昇格したことを確認できます。

### 8.7.2 スイッチオーバー EFM クラスター

PEM クライアントを使用して、フェールオーバーマネージャークラスターのプライマリノードをレプリカノードに置き換えることができます。スイッチオーバープロセスを開始するには、**Tools > Switchover EFM Cluster** を選択します。EFM クラスターをスイッチオーバーすることを確認するプロンプトが表示されます。以下を行うには **Yes** を選択します

- フェールオーバーマネージャーのスイッチオーバーを開始します。

- レプリカノードを読み取り/書き込みプライマリノードの役割に昇格させます。
- プライマリデータベースをフェールオーバーマネージャークラスター内の新しいレプリカとして再構成します。

最も高いプロモーション優先度フェールオーバーマネージャーで定義されたノードが、新しいプライマリノードになります。PEM はジョブステータスを報告します。ジョブが完了し、Streaming Replication Analysis ダッシュボードが更新されたら、Failover Manager ノードのステータステーブルを確認して、スイッチオーバーが発生したことを確認できます。

## 8.8 Monitoring Replication Server

xDB レプリケーションシナリオの一部である EDB Postgres Advanced Server または PostgreSQL データベースから統計を取得するように PEM を構成する前に、xDB レプリケーションを手動でインストールして構成する必要があります。xDB レプリケーションソリューションとドキュメントの詳細については、Multi-master replication operation を参照してください。

PEM xDB Replication プローブは、EDB Postgres Advanced Server または PostgreSQL パブリケーションデータベースがある xDB マルチプライマリまたはシングルプライマリレプリケーションを使用するクラスターの遅延データを監視します。他の独自のデータベースホスト Oracle または SQL Server と EDB Postgres Advanced Server または PostgreSQL 間でレプリケーションを構成している場合、プローブは遅延情報を返すことができません。

デフォルトでは、xDB Replication プローブは無効になっています。 xDB Replication プローブを有効にするには

1. サーバーの名前を右クリックし、コンテキストメニューから **Connect** を選択します。
2. プロンプトが表示されたら、認証情報を入力します。
3. 接続した後、ツリーのサーバーノードを展開し、レプリケートデータベースの名前を選択します。
4. **Management > Manage Probes** を選択します。

**Manage Probes** タブを使用して、xDB Replication プローブを構成します。

- プローブの実行間の分と秒を変更するには、 **Default** を **No** に設定します。
- **Enabled?** スライダーを使用して、xDB Replication プローブを有効にします。
- **Data Retention** フィールドの **Default** スライダーを **No** に設定して、PEM がプローブによって取得された情報を保存する日数を変更します。

プローブを有効にした後、返されたメトリックを使用して、PEM クライアントでカスタムグラフとダッシュボードを作成できます。

## 8.9 Monitoring event history

`pem.event_history` テーブルで実行されたイベントの詳細をモニタできます。ユーザー名、イベントの実行時間、発生したイベントのタイプ、イベントに関する詳細を提供します。テーブルのフィールドは次のとおりです。

| Field name | Description | Example |
|---|---|---|
| recorded_time | イベントの実行時間を記録します。日付と時刻を小数部秒とともに表示します。 | *2025 - 11 - 27 09:14:32* |
| user_name | イベントを実行またはスケジュールしたユーザーのユーザー名。 | *alice@company.com* |
| component | 関連する特定の PEM 機能。 | *profile / alert_blackout / HA Monitoring.* |
| operation | イベントに対して実行されたアクション。 | *assign / disable_alert_blackout / Switchover.* |
| message | イベントに対して実行されたアクションの説明。 | *Enabled the alert blackout for the server.* |
| details | 発生したイベントの JSON 出力。 | *{"ProfileId":42,"ServerId":10}* |

**注釈**

この機能を使用するために JSON の専門家である必要はありません。 **details**フィールドは、実行された特定のアクションに基づいて自動的に調整される「追加情報」のコレクションと考えてください。

### 8.9.1 なぜ重要なのでしょうか？

この `event_history` テーブルは、主要な操作の質問にすばやく回答するのに役立ちます。

- 誰がアラートブラックアウトをオンまたはオフにしましたか？
- 監視プロファイルはいつ適用、削除、または変更されましたか?
- HA クラスターはリーダーを切り替えましたか？いつ、どのサーバー間で？

### 8.9.2 コア特典

- **トラブルシューティング**欠落したアラートがアクティブな停電によるものであったかどうかをすばやく特定します。
- **構成の追跡**最近サーバーに適用されたプロファイルを正確に表示します。
- **HA 可視性**業務時間外に発生したロールの変更とフェイルオーバーを監査します。
- **アカウンタビリティ**変更の背後にある特定のユーザーまたはシステムプロセスを特定します。

### 8.9.3 サポートされている機能

イベント履歴は、現在、次の PEM 機能の変更とアクションを追跡しています。

#### 8.9.3.1 プロファイル管理

- **プロファイルの割り当て**構成プロファイルがサーバーまたはエージェントに割り当てられる場合。
- **プロファイルの削除** プロファイルの割り当てが削除されたとき。
- **プロファイルの更新**プロファイル設定またはカテゴリが変更されたとき。
- **プロフィールの削除**下書きまたは公開されたプロフィールが削除されたとき。
- **プロファイルの公開** ドラフトプロファイルが運用用に公開されたとき。
- **クローンプロファイル** 既存のプロファイルが UI または REST API を介してクローンされた場合。
- **Export Profile** 1 つ以上のプロファイルが `.pemprf` バンドルとしてエクスポートされる場合。 `details` フィールドは、エクスポートされたプロファイルの数とその名前を記録します。
- **インポートプロファイル新しい名前** `.pemprf` バンドルからのプロファイルがまだ存在しない名前でインポートされる場合。`details` フィールドは、新しいプロファイル名とその割り当てられた ID を記録します。
- **インポートプロファイル上書き** `.pemprf` バンドルからプロファイルが既存のプロファイルを上書きしてインポートされ、ドラフトを作成する場合。`details` フィールドは、既存のプロファイル ID と新しいドラフト ID を記録します。

**注釈**

スキップおよび失敗したインポート結果は監査されません。成功したインポート操作のみが `pem.event_history` に書き込まれます。

#### 8.9.3.2 アラートブラックアウト管理

`user_name` 列は、アクションを実行した人を示します。実際のログインたとえば `enterprisedb` とは、UI または REST API を介して操作されたユーザーを意味します。値 `system` は、システムジョブがアクションを自動的に実行したことを意味します。

| Operation | Trigger |
|---|---|
| *enable_alert_blackout* | Alert Blackout checkbox toggled on, or *alert_blackout* flag synced by the **Process alert blackouts** system job |
| *disable_alert_blackout* | Alert Blackout checkbox toggled off, or *alert_blackout* flag synced by the **Process alert blackouts** system job |
| *schedule_alert_blackout* | Scheduled blackout created via the Schedule Alert Blackout dialog |
| *delete_scheduled_blackout* | Scheduled blackout deleted via the Schedule Alert Blackout dialog |
| *create_blackout* | Blackout created via REST API POST |
| *end_blackout* | Blackout ended via REST API DELETE |
| *expire_manual_blackout* | Manual blackout auto-expired because *max_manual_blackout* was exceeded |
| *blackout_ended* | Checkbox uncheck ended an active scheduled or auto blackout early |
| *blackout_completed* | Scheduled blackout naturally completed when its end time was reached |
| *auto_enable_blackout* | Auto blackout created for an unreachable server or agent |

次のページに続く

表 45 – 前のページからの続き

| Operation | Trigger |
|---|---|
| *auto_disable_blackout* | Auto blackout cleared after the server or agent recovered |

#### 8.9.3.3 高可用性 HA モニタリング

- **スイッチオーバー/フェールオーバー** HA クラスタープライマリ↔レプリカでの計画的または計画外のロールの変更。

記録された各イベントには以下が含まれます。

- 発生したときの **タイムスタンプ** 。
- アクションまたは自動イベントの場合は `system` を開始した**ユーザー** 。
- **影響を受けるリソース**サーバー ID、エージェント ID、クラスター名など。
- **以前の値と新しい値** 該当する場合。
- JSON 詳細フィールドに保存される **追加コンテキスト** 。

### 8.9.4 一般的な運用シナリオ

次のエントリは `event_history` ログにキャプチャされます。

- **プロファイルの割り当て**監視プロファイル実稼働 CPU とメモリをサーバーに適用します。ログは、ユーザー、プロファイル ID、および構成が適用された正確な時間をキャプチャします。
- **アラートブラックアウト**サーバーまたはエージェントのアラートブラックアウトを有効にします。これにより、特定のメンテナンス期間に通知が抑制された理由の明確な監査レコードが提供されます。
- **高可用性 HA 移行**プライマリロールを引き継ぐスタンバイノード。イベントはロールの変更を文書化し、関係するサーバーの以前のロールと現在のロールの両方を特定します。
- **管理上のクリーンアップ**古いまたは未使用のドラフトプロファイルの削除。削除の永続的な記録は、完全な履歴論文の証跡を保証のために維持されます。

### 8.9.5 イベントエントリーリファレンス

以下のセクションでは、さまざまなアクティビティがログ内に記録される方法の内訳を提供します。

#### 8.9.5.1 1. プロファイルの割り当て

**メッセージ** “サーバー 10 にプロファイル 42 を割り当てます”。

監視プロファイル ID 42 がサーバー ID 10 に新しく適用されると、ログは次のエントリを生成します。

**``details`` (JSON):**

```
{
  "ProfileId": 42,
  "PreviousProfileId": null,
  "ServerId": 10,
  "TargetType": "server",
  "ChangedCategories": ["assigned_profile"]
}
```

#### 8.9.5.2 2. プロファイルのクローン

**メッセージ** 「クローンプロファイル 42」。

プロファイル ID 42 が UI **Copy from** ダイアログまたは REST API を介してクローン化されると、両方のパスが同じ監査行を書き込みます。監査履歴でクローンイベントを検索するには、`operation = 'clone_profile'` でフィルタリングします。通常の非クローンプロファイル作成は、代わりに `operation = 'create'` を使用します。

**``details`` (JSON):**

```
{
  "SourceProfileId": 42,
  "SourceName": "Production Server Profile",
  "NewProfileId": 55,
  "NewName": "Copy of Production Server Profile",
  "TargetKind": "server",
  "ChangedCategories": ["metadata", "probe", "alert"]
}
```

**注釈**

監査ログはベストエフォートです。監査行の書き込みに失敗した場合、クローンは成功します。障害はサーバー側のログに記録されます。

#### 8.9.5.3 3. プロファイルのエクスポート

**メッセージ** “エクスポートされた 2 つのプロファイル: 実稼働サーバープロファイル、ステージングサーバープロファイル”。

1 つ以上のプロファイルが `.pemprf` バンドルとしてエクスポートされると、ログには、エクスポートされたプロファイルの数と名前が記録されます。

**``details`` (JSON):**

```
{
  "Count": 2,
  "ExportedNames": ["Production Server Profile", "Staging Server Profile"]
}
```

#### 8.9.5.4 4. プロファイルのインポート新しい名前

**メッセージ** 「インポートされたプロファイル ‘Staging Server Profile’」。

`.pemprf` バンドルからのプロファイルがターゲットインスタンスにまだ存在しない名前でインポートされると、新しいプロファイルの名前と割り当てられた ID がログに記録されます。

**``details`` (JSON):**

```
{
  "Name": "Staging Server Profile",
  "ProfileId": 67
}
```

#### 8.9.5.5 5. プロファイルインポート上書き → 下書き

**メッセージ**「既存のプロファイル 42 の上にドラフトとしてプロファイル’Production Server Profile’をインポートしました」。

インポートが既存のプロファイルを名前で上書きする場合 **Skip stable?** のチェックを外して、インポートされた構成はドラフトとして保存されます。ログには、既存のプロファイル ID と新しいドラフト ID の両方が記録されます。

**``details`` (JSON):**

```
{
  "ExistingProfileId": 42,
  "DraftId": 68
}
```

> **注釈**
>
> スキップされ失敗したインポート結果は、監査エントリを生成しません。成功したインポートのみが `pem.event_history` に記録されます。

#### 8.9.5.6 6. アラートブラックアウト有効チェックボックス

**メッセージ** “サーバーのアラートブラックアウトを有効にしました”。

ユーザーがグローバル概要ダッシュボードの Alert Blackout チェックボックスをチェックしたときに記録されます。

**``details`` (JSON):**

```
{
  "IsAgent": false,
  "ObjectId": 10
}
```

#### 8.9.5.7 7. アラートブラックアウトは無効チェックボックス

**メッセージ** “サーバーのアラートブラックアウトを無効にしました”。

ユーザーが Alert Blackout チェックボックスのチェックを外したときに記録されます。チェックを解除すると、オブジェクトのアクティブなブラックアウトがすべて終了するため、EndedBlackouts は、クローズされたすべてのレコード、そのソース、およびスケジュールされた終了時間より前に終了したかどうかをリストします。

**``details`` (JSON):**

```
{
  "IsAgent": false,
  "ObjectId": 10,
  "EndedBlackouts": [
    { "Id": 42, "Source": "manual", "ScheduledEnd": null, "EndedEarly": false },
    { "Id": 43, "Source": "scheduled", "ScheduledEnd": "2026-04-24T02:00:00+00:00",
↪"EndedEarly": true }
  ],
  "EndedCount": 2,
  "EndedSources": ["manual", "scheduled"]
}
```

#### 8.9.5.8 8. スケジュールされた停電が作成されましたダイアログ

**メッセージ** “スケジュールされたアラートブラックアウトを作成しました。”

スケジュールされたブラックアウトが [ Schedule Alert Blackout ] ダイアログを介して保存されたときに記録されます。 BlackoutGroups は、バッチ内の各グループのオブジェクト、開始時間、および期間をキャプチャします。

**``details`` (JSON):**

```
{
  "BlackoutGroups": [
    {
      "ObjectType": "server",
      "ObjectIds": [5, 7],
      "StartTime": "2026-04-23T10:00:00Z",
      "Duration": "2 hours"
    }
  ],
  "CreatedCount": 2
}
```

#### 8.9.5.9 9. 計画的な停電の削除ダイアログ

**メッセージ**「スケジュールされたアラートのブラックアウトを削除しました。」

スケジュールされたブラックアウトが [ Schedule Alert Blackout ] ダイアログを介して削除されたときに記録されます。

**``details`` (JSON):**

```
{
  "DeletedIds": ["a1b2c3d4-e5f6-7890-abcd-ef1234567890"],
  "AgentIds": [],
  "ServerIds": [5, 7]
}
```

#### 8.9.5.10 10. 手動ブラックアウトの有効期限が切れました

**メッセージ**“手動ブラックアウトの有効期限が切れました max_manual_blackout を超えました”。

手動ブラックアウトが `max_manual_blackout` しきい値より長くアクティブになったときに、**プロセスアラートブラックアウト** システムジョブによって記録されます。`user_name` は `system` です。

**``details`` (JSON):**

```
{
  "IsAgent": false,
  "Ids": "10",
  "MaxManualBlackoutHours": 24
}
```

#### 8.9.5.11 11. 計画的な停電が完了しました

**メッセージ**「計画的な停電が完了しました。」

スケジュールされた停電が自然に終了時間に到達すると、システムジョブによって記録されます。`user_name` は `system` です。

**``details`` (JSON):**

```
{
  "IsAgent": false,
  "Ids": "10",
  "Source": "scheduled",
  "BatchId": "a1b2c3d4-e5f6-7890-abcd-ef1234567890"
}
```

#### 8.9.5.12 12. 自動ブラックアウトが有効になっています

**メッセージ**「到達不能なサーバーのオートブラックアウトが有効になっています」。

サーバーまたはエージェントが server_contact_timeout しきい値を超えると、 **Blackout unreachable servers/agents** システムジョブによって記録されます。 user_name は system です。

**``details`` (JSON):**

```
{
  "IsAgent": false,
  "Ids": "10",
  "Source": "auto"
}
```

#### 8.9.5.13 13. 自動ブラックアウトは無効です

**メッセージ**「復旧したサーバーの自動ブラックアウトはクリアされました。」

以前に到達できなかったサーバーまたはエージェントがタイムアウトウィンドウ内にハートビートを再開したときに、システムジョブによって記録されます。user_name は system です。

**``details`` (JSON):**

```
{
  "IsAgent": false,
  "Ids": "10",
  "Source": "auto"
}
```

#### 8.9.5.14 14.HA スイッチオーバー

**メッセージ**“クラスター prod_patroni のスイッチオーバーが観察されました”。

ノードロールの変更をログに記録します。これは、高可用性イベントとフェールオーバーイベントの監査に重要です。

**``details`` (JSON):**

```
{
  "ClusterType": "patroni",
  "ClusterName": "prod_patroni",
  "OldRole": "primary",
  "NewRole": "replica",
  "NodeIp": "10.2.3.15"
}
```

### 8.9.6 最近のアクティビティを一目で見る方法

PEM 内の SQL クエリウィンドウからこれらのクエリを実行して、最近の変更をすばやく監査できます。これらの操作は読み取り専用で、データは変更されません。

#### 8.9.6.1 最近のプロフィールイベント

このクエリを使用して、構成プロファイル割り当て、更新、または削除に関連する最新の 20 件のアクションを表示します。

```
SELECT recorded_time, user_name, message
FROM pem.event_history
WHERE component = profile
ORDER BY recorded_time DESC
LIMIT 20;
```

#### 8.9.6.2 最近のプロファイルクローンイベント

このクエリを使用して、クローン化されたすべてのプロファイルと、details フィールドからのソースおよび新しいプロファイル ID を表示します。

```
SELECT recorded_time, user_name,
       details::json->>SourceName AS source_profile,
       details::json->>NewName AS cloned_profile
FROM pem.event_history
WHERE component = profile AND operation = clone_profile
ORDER BY recorded_time DESC
LIMIT 20;
```

#### 8.9.6.3 最近のアラートブラックアウトの変更

このクエリを使用して、すべてのブラックアウト状態の変更を最新のものから順に表示します。

```
SELECT recorded_time, user_name, operation, message
FROM pem.event_history
WHERE component = alert_blackout
ORDER BY recorded_time DESC
LIMIT 20;
```

特定のサーバーたとえばオブジェクト ID 10 のブラックアウト状態を変更した人を確認するには

```
SELECT recorded_time, user_name, operation, message
FROM pem.event_history
WHERE component = alert_blackout
  AND (
    details::jsonb->>Ids = 10
```

(次のページに続く)

(前のページからの続き)

```
    OR details::jsonb->>ObjectId = 10
    OR details::jsonb->ServerIds @> [10]::jsonb
  )
ORDER BY recorded_time DESC;
```

過去 7 日間のすべての手動ブラックアウト有効期限を見つけるには

```
SELECT recorded_time, details
FROM pem.event_history
WHERE component = alert_blackout
  AND operation = expire_manual_blackout
  AND recorded_time > now() - interval 7 days;
```

スケジュールされたブラックアウトバッチの完全なライフサイクルを表示するには

```
SELECT recorded_time, operation, message, details
FROM pem.event_history
WHERE component = alert_blackout
  AND details::jsonb->>BatchId = <batch-id>
ORDER BY recorded_time;
```

自動ブラックアウトアクティビティを監査するには到達不能なサーバーとエージェント

```
SELECT recorded_time, operation, message, details
FROM pem.event_history
WHERE component = alert_blackout
  AND operation IN (auto_enable_blackout, auto_disable_blackout)
ORDER BY recorded_time DESC;
```

#### 8.9.6.4 最近の HA スイッチオーバー

このクエリを使用して、HA クラスターがリーダープライマリ/レプリカロールを変更したときを特定します。

```
SELECT recorded_time, message
FROM pem.event_history
WHERE component = HA Monitoring AND operation = Switchover
ORDER BY recorded_time DESC
LIMIT 10;
```

### 8.9.7 詳細フィールドの読み取り

ProfileId などの JSON データから特定の値を抽出する必要がある場合は、Postgres JSON 演算子を使用して自分の列に直接取り出すことができます。

```
SELECT recorded_time,
       details::json->>ProfileId AS profile_id,
       message
FROM pem.event_history
WHERE component = profile
ORDER BY recorded_time DESC
LIMIT 10;
```

**結果の.. Note::**

*NULL* 値は、これらの特定のイベントの JSON データにその特定のキーが含まれていないことを示します。

### 8.9.8 考慮事項

- **変更スパイクの監視**ブラックアウトトグルまたはプロファイルの変更の突然の増加は、構成の混乱または環境の不安定を示している可能性があります。
- **意味のある名前を使用します**プロフィールに明確で説明的な名前を割り当てます。これにより、ID のみに依存するのではなく、message 列が一目で読みやすくなります。
- **データのメンテナンス**テーブルが非常に大きくなった場合、古い行を定期的にエクスポートまたはアーカイブします。最適なローカルポリシーについては、管理者に相談してください。
- **整合性の維持**常にテーブルを永続的な履歴として扱います。既存の行を手動で編集または変更することは避けてください。
- **イベントの削除と監査**古いイベントを削除することは可能ですが、監査証跡が削除されるためお勧めしません。代わりに、古いデータをセカンダリ記憶場所にアーカイブして、テーブルサイズを管理しながら履歴を維持します。
- **システム生成イベント** unknown ユーザーをリストするイベントは、通常、PEM が特定のユーザーアカウントを識別できなかったことを示します。これは一般的に、システム生成タスクまたは古い移行スクリプトで発生します。
- **イベントデータの解釈** message 列は、プライマリイベントの詳細をわかりやすい英語で提供します。詳細な分析のために特定の低レベルの技術的な詳細が必要な場合にのみ、JSON データを解析する必要があります。

## 8.10 Tuning performance

次の機能は、パフォーマンスの調整に役立ちます。

- *Performance Diagnostic*

### 8.10.1 Performance Diagnostic

パフォーマンス診断ダッシュボードは、待機イベントを監視することにより、Postgres インスタンスのデータベースパフォーマンスを分析します。診断グラフを表示するために、PEM は EDB 待機状態モジュールによって収集されたデータを使用します。EDB 待機状態の詳細については、 EDB wait states を参照してください。

複数のレベルで待機状態データを分析するには、選択したデータを絞り込みます。グラフの上位レベルで選択したデータは、下位レベルに設定されます。

> **注釈**
>
> パフォーマンス診断ダッシュボードは、EDB 待機状態モジュールが利用可能なすべての Postgres バリアントとバージョンで利用できます。

#### 8.10.1.1 前提条件

- EDB 待機状態パッケージのインストール
- PostgreSQL の場合は、 EDB repositories を参照してください .
- EDB Postgres Advanced Server の場合は、EDB wait states を参照してください。
- EDB Postgres Advanced Server の EDB 待機状態モジュールをインストールした後

1. `postgresql.conf` ファイルにライブラリのリストを構成します。

```
shared_preload_libraries = $libdir/edb_wait_states
```

2. データベースサーバーを再起動します。 3. メンテナンスデータベースに次の拡張機能を作成します。

```
CREATE EXTENSION edb_wait_states;
```

- スーパーユーザー権限のないユーザーとしてパフォーマンス診断ダッシュボードにアクセスする場合は、ユーザーが接続またはログインしているサーバーに基づいて、特定の **ログインロール** をユーザーに付与する必要があります。
- PEM サーバーにログインしているユーザーこの例では `test1` に `pem_comp_performance_diagnostic` ロールを付与します。

```
GRANT pem_comp_performance_diagnostic TO test1;
```

- `edb_wait_states` のすべての機能に対する `pg_monitor` ロールと `EXECUTE` 特権をデータベースユーザーこの例では `test2` に付与します。 `test2` ユーザーは、EDB 待機状態でデータベースサーバーに接続しています。

```
# Grant `pg_monitor` role
GRANT pg_monitor TO test2;

# Grant `Execute` privilege on all functions
```

(次のページに続く)

(前のページからの続き)

```
GRANT EXECUTE ON FUNCTION edb_wait_states_sessions TO test2;
GRANT EXECUTE ON FUNCTION edb_wait_states_data TO test2;
GRANT EXECUTE ON FUNCTION edb_wait_states_queries TO test2;
GRANT EXECUTE ON FUNCTION edb_wait_states_samples TO test2;
GRANT EXECUTE ON FUNCTION edb_wait_states_purge TO test2;
```

前提条件が満たされていない場合、パフォーマンス診断ダッシュボードにアクセスするとエラーが表示されます。

#### 8.10.1.2 パフォーマンス診断ダッシュボードを使用する

Performance Diagnostic ダッシュボードを開くには、PEM クライアントで **Tools > Server > Performance Diagnostic** を選択します。

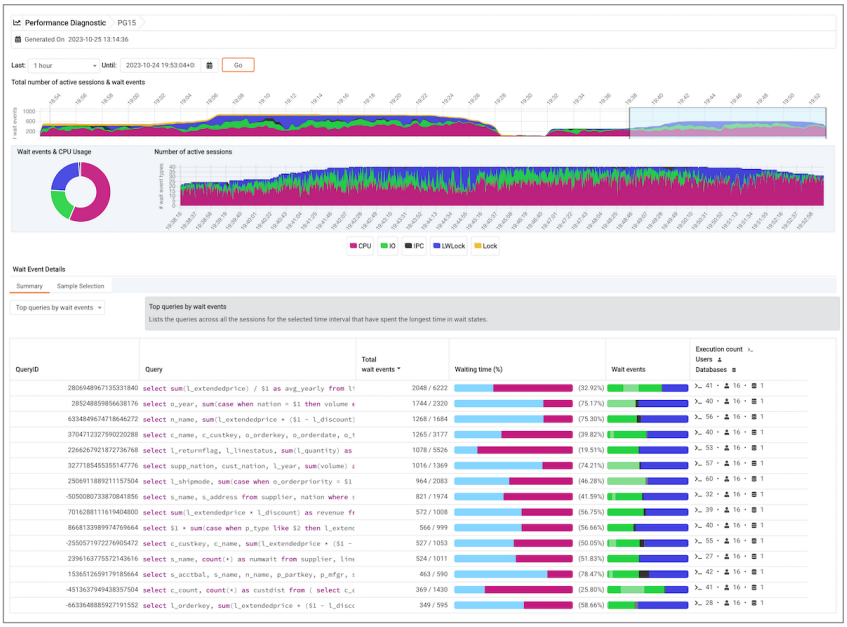



図 14: Performance Diagnostic dashboard

デフォルトでは、上部のパフォーマンス診断グラフは、現在の日時から開始して、過去 1 時間のデータを取得します。このグラフは、アクティブなセッションの数を含む時系列を示しています。この時系列の各ポイントは、過去 15 秒間の待機イベントと CPU 使用量の合計を表します。これらのセッションは、wait イベントを待機しているか、特定の時点で CPU を使用している可能性があります。この時系列は、EDB Wait States 拡張機能によって収集された待機イベントサンプルに基づいて生成されます。

新しいブラウザータブでパフォーマンス診断ダッシュボードを開くには、**Preferences > Open in New Browser Tab?** を選択します。

##### パフォーマンス診断ダッシュボード

- 最初のグラフには、選択した期間でアクティブなセッションの数と待機イベントタイプが表示されます。最初のグラフの期間を変更して、特定の期間のデータを分析できます。

デフォルトでは、最初のグラフの期間選択は 15 分です。持続時間は 1 時間まで延長できます。グラフで期間を表示するには、**Last** リストから期間を選択します。指定した日付と時刻のデータを表示するには、**Until** リストから日付と時刻を選択します。

- 次のセクションでは、最初のグラフで選択した期間に基づいて、次のグラフをプロットします。
- ドーナツグラフ - 最初のグラフでの期間選択に応じて、待機イベントタイプの合計と CPU 使用率を表示します。このグラフは、これらのセッションがイベントの待機に費やした時間をより深く理解できます。
- 折れ線グラフ - 各ポイントがサンプル時間のアクティブなセッションを表す時系列をプロットします。

各待機イベントタイプと CPU 使用率をより明確に区別するために、各待機イベントタイプのグラフは別の色で表示されます。

折れ線グラフの時間を選択して、待機イベントを分析します。すべてのグラフで待機イベントタイプを表示または非表示にするには、グラフの凡例を選択します。分析する待機イベントタイプのみを表示できると、分析が簡単になります。

- 3 番目のセクションには、2 番目のグラフで選択した期間に基づいて、パフォーマンス診断ダッシュボードに待機イベントの詳細が表示されます。2 つのタブに待機イベントの詳細が表示されます。
- **Overview** タブには、クエリ ID、合計 CPU イベント、CPU 使用率、待機イベント、および選択した時間間隔の実行数を含む上位 SQL クエリのリストが表示されます。これらの SQL クエリーは 4 つのカテゴリにグループ化されています。
- **待機イベント別の上位クエリー** - 待機状態で最も長い時間を費やした選択したサンプル時間のすべてのセッションのクエリーをリストします。クエリ ID、クエリ、合計待機イベント、パーセンテージでの待機時間、待機イベント、および実行数と、そのクエリを実行したユーザーの数がデータベースの数で表示されます。
- **CPU 使用率別の上位クエリ**-選択したサンプル時間のすべてのセッションで、最も CPU 時間を使用したクエリーをリストします。クエリ ID、クエリ、合計 CPU イベント、CPU 使用率、待機イベント、および実行数と、データベースの数でそのクエリを実行したユーザーの数が表示されます。テーブルのデータは、CPU イベントの数の降順です。
- **待機イベント別のトップユーザー** - 選択したサンプル時間のすべてのセッションで、クエリが待機状態で最も長い時間を費やしたユーザーをリストします。ユーザー名、データベース名、合計待機イベント、待機時間の割合、待機イベント、および待機状態で最も長い時間を費やしたクエリーの数が表示されます。
- **CPU 使用率によるトップユーザー** - すべてのユーザーのリストクエリが最も多くの CPU 時間を使用した選択したサンプル時間のセッション。ユーザー名、データベース名、合計 CPU イベント、CPU 使用率、待機イベント、および最も多くの CPU 時間を使用したクエリーの数が表示されます。
- **Sample Selection** タブには、選択したものの待機イベントの詳細が表示されます。2 番目のグラフのサンプル時間。待機イベントは次のようにグループ化されています。
- **Queries** - クエリ ID と選択したサンプル時間にクエリを実行したセッション数とともにクエリをリストします。

任意の行の目を選択して、そのクエリに関する詳細を示すウィンドウを表示します。詳細については、クエリー情報 を参照してください。

- **Users** - クエリを実行したユーザーによってグループ化された待機イベントの詳細と、クエリのイベント数と実行数をリストします。
- **Waits** - リスト選択したサンプル時間の各待機イベントタイプに属する待機イベントの数。

両方のタブの下の行に表示されるデータをフィルタリングできます。列ヘッダーを選択して、データをアルファベット順に並べ替えることもできます。

### クエリー情報

クエリ情報ウィンドウを開くには、 **Sample Selection** タブの下の **Queries** グループの任意の行で目を選択します。このウィンドウには、**Query information** セクションに、選択したサンプル時間でのクエリ ID とそれに対応するセッション ID がリストに表示されます。分析するデータを含む選択したクエリの**セッション ID** リストを選択できます。選択したセッション ID とクエリ ID に対応する詳細が表示されます。**クエリ情報** テーブルには、 SQL クエリーも表示されます。

**Wait Event locations** セクションには、選択したセッション ID とクエリ ID の待機イベントタイプの合計数が表示されます。2 種類のグラフが表示されます。

- ドーナツグラフ - カテゴリデータの割合を示します。
- タイムライン棒グラフ - 時間の経過とともに待機イベントタイプの数の傾向を視覚化します。

区別するために、各待機イベントタイプは棒グラフの異なる色で表されます。

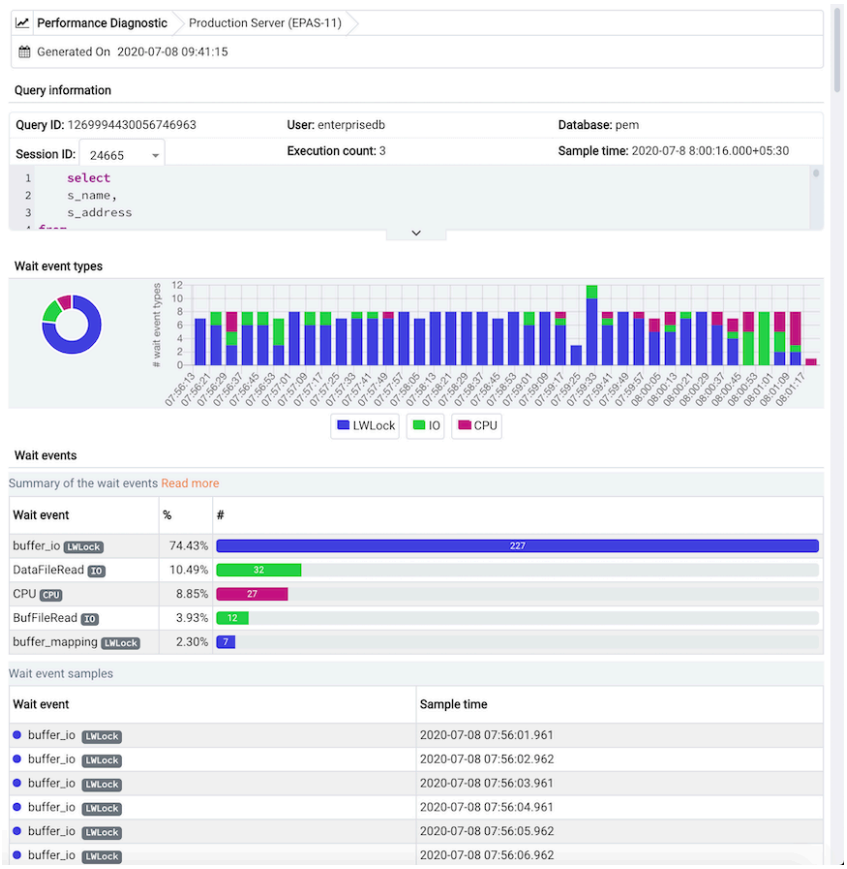



図 15: Performance Diagnostic - Query dashboard

**Wait events** セクションには、クエリの実行中に発生したすべての待機イベントを表示するテーブルがあり

ます。待機イベントの数の降順にデータを表示します。2 番目のテーブルには、クエリ実行全体の期間に発生したサンプル時間と待機イベントが表示されます。これにより、一定の期間にわたってクエリ実行中の待機イベントを分析できます。また、そのクエリ ID とセッション ID の EDB Wait States 拡張機能によって収集された実際のサンプルも表示します。

## 8.11 Profiling workloads

SQL プロファイラー拡張機能は PEM と連携して、サーバーのワークロードを詳細にプロファイルできます。SQL プロファイラーを使用して、クエリプラン、バッファ使用量、実行時間などを収集します。

- *Installing the SQL Profiler extension* を参照
- *Upgrading SQL Profiler* を参照
- *Using SQL Profiler* を参照

### 8.11.1 Installing the SQL Profiler extension

使用する各サーバーに SQL プロファイラー拡張機能をインストールして有効にする必要があります。たとえば、PostgreSQL 14 および PostgreSQL 15 を実行しているホストがある場合は、各サーバーで拡張機能を有効にする必要があります。

SQL プロファイラーは、使用している Postgres ディストリビューションと同じプラットフォームでサポートされています。参照

- EDB Postgres Advanced Server Product Compatibility
- PostgreSQL Product Compatibility
- EDB Postgres Extended Server Product Compatibility

#### 8.11.1.1 パッケージのインストール

**前提条件**

インストールプロセスを開始する前に

- Postgres をインストールします。 - Installing EDB Postgres Advanced Server を参照してください。
- Installing PostgreSQL
- Installing EDB Postgres Extended Server
- リポジトリの設定をします。

リポジトリのセットアップは 1 回限りのタスクです。すでにリポジトリを設定している場合、この手順を実行する必要はありません。

リポジトリを設定するには、EDB repositories に移動し、そこに提供されている指示に従ってください。

#### パッケージをインストールする

install コマンドの構文は次のとおりです。

```
sudo <package-manager> -y install edb-<postgres><postgres_version>-server-sqlprofiler
```

そこで

- <package-manager> は、オペレーティングシステムで使用されるパッケージマネージャーです。

| Package manager | Operating system |
|---|---|
| dnf | RHEL 8 and derivatives |
| zypper | SLES |
| apt-get | Debian and derivatives |

- <postgres> は、使用している Postgres のディストリビューションです。

| Postgres distribution | Value |
|---|---|
| PostgreSQL | pg |
| EDB Postgres Advanced Server | as |
| EDB Postgres Extended | postgresextended |

- <postgres_version> は、使用している Postgres のバージョンです。

たとえば、RHEL 8 プラットフォームに EDB Postgres Advanced Server 15 の SQL プロファイラーの最新バージョンをインストールするには

```
sudo dnf -y install edb-as15-server-sqlprofiler
```

### 8.11.1.2 拡張機能の有効化

拡張機能を有効にするには

1. プロファイルするサーバー上の postgresql.conf ファイルを編集し、shared_preload_libraries パラメーターを変更します。

```
shared_preload_libraries = $libdir/sql-profiler
```

1. Postgres サーバーを再起動します。
2. 次のコマンドを使用して、メンテナンスデータベースに SQL プロファイラー拡張機能を作成します。

```
# Connect to your maintenance database using psql
psql -d postgres -U enterprisedb
# Create the extension
CREATE EXTENSION sql_profiler;
```

> **注釈**
>
> SQL プロファイラーを構成する前に PEM クライアントを使用して PEM サーバーに接続している場合、サーバーを切断して再接続して SQL プロファイラー機能を有効にする必要があります。

拡張機能を有効にすると、サーバーに存在するすべてのデータベースで SQL プロファイラーを使用できるようになります。

## 8.11.2 Using SQL Profiler

SQL プロファイラー拡張機能を使用すると、ユーザーは非効率な SQL コードを見つけて最適化できます。Microsoft の SQL Server プロファイラーは、操作と機能が PEM の SQL プロファイラーに非常に似ています。

SQL プロファイラーは PEM と連携して、サーバーのワークロードをプロファイリングできます。PEM エージェントの有無にかかわらずサーバーに SQL プロファイラー拡張機能をインストールして有効にできます。ただし、トレースは非管理対象サーバーでアドホックモードでのみ実行でき、管理対象サーバーでのみスケジュールできます。

SQL プロファイラーは、SQL トレースでの分析のために特定の SQL ワークロードをキャプチャおよび表示します。キャプチャされた SQL のトレースをすぐに開始して確認するか、キャプチャされたトレースを保存して後で確認できます。SQL プロファイラーを使用して、最大 15 の名前付けトレースを作成および保存できます。

### 8.11.2.1 SQL プロファイラーの権限

PEM で SQL プロファイラーツールにアクセスするには、2 つの前提条件があります。

1. PEM GUI にログインしているユーザーは、スーパーユーザーまたは PEM サーバーデータベース内のグループ `pem_comp_sqlprofiler` のメンバーである必要があります。詳細については、

pem groups を参照してください。

2. サーバーツリー `Username` でデータベースサーバー用に構成されたユーザーは、トレースが実行されているデータベースサーバー監視対象サーバーのスーパーユーザーである必要があります。

SQL プロファイラーバージョン 4.2.0 以降では、`sql_profiler_admin` ロールにより、非スーパーユーザーが SQL プロファイラーファンクションを実行できます。このロールをユーザーに割り当てるには、次のコマンドを実行します。

```
grant sql_profiler_admin to username
```

### 8.11.2.2 トレースの作成

[ Create Trace ] ダイアログボックスを使用して、SQL プロファイラーがインストールおよび構成されているデータベースの SQL トレースを定義できます。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データベースを選択し、**Tools > Server > SQL Profiler > Create Trace** を選択します。

**Trace options** タブを使用して、新しいトレースに関する詳細を指定します。

- **Name** フィールドにトレースの名前を入力します。

- **User filter** フィールドを使用して、トレースにクエリを含めるロールを指定します。すべてのロールからのクエリを含めるには、**Select All** を選択します。
- **Server Filter** フィールドを使用して、トレースするデータベースを指定します。すべてのデータベースに対するクエリーを含めるには、**Select All** を選択します。
- **最大トレースファイルサイズ**フィールドにトレースサイズを指定します。SQL プロファイラーは、指定されたサイズにほぼ達すると、トレースを終了します。
- **Create** を選択したときにトレースを開始するには、 **Run Now** フィールドで **Yes** を選択します。 [ **Schedule** ] タブのフィールドを有効にするには、 **No** を選択します。

**Schedule** タブを使用して、新しいトレースのスケジュールの詳細を指定します。

- **Start time** フィールドを使用して、トレースの開始時間を指定します。
- **End time** フィールドを使用して、トレースの終了時間を指定します。
- 毎日指定された時間にトレースを繰り返すには、**Repeat?** フィールドで **Yes** を選択します。 [ **Periodic job options** ] タブのフィールドを有効にするには、 **No** を選択します。

**Periodic job options** タブを使用して、定期的なトレースに関するスケジュールの詳細を指定します。**Days** セクションを使用して、ジョブを実行する日を指定します。

- **Week days** フィールドを使用して、トレースを実行する曜日を選択します。
- **Month days** フィールドを使用して、トレースを実行する日を選択します。
- **Months** フィールドを使用して、トレースを実行する月を選択します。

**Times** セクションを使用して、トレース実行のスケジュールを指定します。**Hours** および **Minutes** フィールドを使用して、トレースを実行する時間を指定します。

[ Create Trace ] ダイアログボックスの完了後、[ **Create** ] を選択してトレースを開始またはトレースをスケジュールします。

トレースをすぐに実行すると、トレース結果が PEM クライアントに表示されます。

#### 8.11.2.3 既存のトレースを開く

以前のトレースを表示するには、PEM クライアントツリーでプロファイルされたデータベースを選択し、**Management > SQL Profiler > Open Trace** を選択します。「 Open Trace 」ダイアログボックスが開きます。

トレースリストからエントリを選択し、 **Open** を選択して、選択したトレースを **SQL Profiler** タブで開きます。

#### 8.11.2.4 トレースのフィルタリング

フィルタは、1 つ以上のルールの名前付けセットであり、それぞれがトレースビューからイベントを非表示にできます。トレースにフィルタを適用すると、非表示のイベントはトレースから削除されませんが、表示から除外されます。

**Filter** を選択して [ Trace Filter ] ダイアログボックスを開き、フィルタを定義するルールまたはルールセットを作成します。各ルールは、イベントを呼び出したロールの ID またはイベント中に呼び出されたクエリタイプに基づいて、現在のトレース内のイベントをスクリーニングします。

既存のフィルタを開くには、 **Open** を選択します。新しいフィルタを定義するには、 **Add (+)** を選択して [ **General** ] タブに表示されるテーブルに行を追加し、ルールの詳細を提供します。

- **Type** リストを使用して、フィルタルールを適用するトレースフィールドを指定します。
- **Condition** リストを使用して、トレースをフィルタリングするときに値に適用する SQL プロファイラーの演算子の種類を指定します。
  - 指定された値を含むイベントをフィルタリングするには、 **Matches** を選択します。
  - **Does を選択します not match** 指定された値を含まないイベントをフィルタリングします。
  - **Value** フィールドで指定された文字列との完全な一致を含むイベントをフィルタリングするには、**Is 等しい** を選択します。
  - **を選択します** Value **フィールドで指定された文字列との完全な一致を含まないイベントをフィルタリングするには、**と等しくない**-で指定された文字列で始まるイベントをフィルタリングするには、Starts with を選択します Value** フィールド.
  - **Value** フィールドで指定された文字列で始まらないイベントをフィルタリングするには、**Does not start with** を選択します。
  - **Less than** を選択して、次のイベントをフィルタリングします **Value** フィールドで指定された数より小さい数値。
  - nu を持つイベントをフィルタリングするには **より大きい** を選択します **Value** フィールドで指定された数より大きいメトリック値。
  - **Value** で指定された数以下の数値を持つイベントをフィルタリングするには、**Less than or 等しい** を選択します field.
  - **Value** フィールドで指定された数以上の数値を持つイベントをフィルタリングするには、**Greater than or equal to** を選択します。
- **Value** フィールドを使用して、検索する文字列、数値、または正規表現を指定します。

ルールの定義が終了したら、 **Add (+)** を選択して別のルールをフィルタに追加します。フィルタからルールを削除するには、ルールを選択し、**Delete** を選択します。

フィルタを適用せずにフィルタ定義をファイルに保存するには、 **Save** を選択します。フィルタを適用するには、 **OK** を選択します。

#### 8.11.2.5 トレースの削除

トレースを削除するには

1. PEM クライアントツリーでプロファイルされたデータベースを選択します。
2. **Management > SQL Profiler > Delete Trace(s)** を選択します。
3. Delete Traces ダイアログボックスで、トレース名の左側にあるアイコンを選択して、削除する 1 つ以上のトレースをマークします。
4. **削除** を選択します。PEM クライアントは、選択されたトレースが削除されたことを確認します。

#### 8.11.2.6 スケジュールされたトレースの表示

スケジュールされたトレースのリストを表示するには、PEM クライアントツリーでプロファイルされたデータベースを選択します。**Management > SQL Profiler > Scheduled Traces** を選択します。

[ Scheduled Traces ] ダイアログボックスには、実行待機中のトレースのリストが表示されます。トレース名の左側にある **Edit** を選択して、トレースに関する詳細情報を表示します。

- **Status** フィールドは、現在のトレースのステータスをリストします。
- **Enabled?** スイッチは、トレースが有効になっている場合は Yes、無効になっている場合は No と表示されます。
- **Name** フィールドには、トレースの名前が表示されます。
- **Agent** フィールドには、トレースを実行するエージェントの名前が表示されます。
- **Last run** フィールドには、トレースが最後に実行された日時が表示されます。
- **Next run** フィールドには、次にスケジュールされたトレースの日時が表示されます。
- **Created** フィールドには、トレースが定義された日時が表示されます。

## 8.12 Reports

監視対象サーバーの診断データを管理および確認します。 Postgres Enterprise ManagerPEM は、診断ニーズに合わせて 2 つの異なるレポートフレームワークを提供します。

### 8.12.1 Custom report templates

**Custom Reports** フレームワークを使用すると、オーダーメイドの診断テンプレートをデザインおよび実行できます。SQL クエリーとホストレベルのスクリプトを組み合わせることで、固有のインフラストラクチャ要件に合わせて調整された特定の分析情報を抽出できます。

- **ユーザー定義ロジック** - SQL またはシェル/バッチスクリプトを使用して、テンプレートをゼロからビルドします。
- **階層構造** - 明確なレポートのために、データを論理セクションと項目に整理します。
- **移植性** - JSON を介してテンプレートをエクスポートおよびインポートし、環境間で簡単に移行できます。

### 8.12.2 Viewing system reports

**システムレポート**は、PEM が提供する事前定義されたビルトイン診断ツールです。これらのレポートは、データベースパフォーマンス、セキュリティ監査、および構成状態のすぐに使える分析を提供します。

- **標準化されたメトリック** - データベースの状態とアクティビティに関する業界標準のレポートにアクセスします。
- **構成は必要ありません** - カスタムスクリプトを書く必要がなく、即時診断を実行します。
- **深い統合** - PEM のアラートおよびモニタリング履歴と完全に統合されています。

> **注釈**
>
> レポートを実行するには、PEM ブラウザツリーで**ターゲットサーバー**を右クリックし、関連するレポートカテゴリに移動するだけです。

#### 8.12.2.1 Custom report templates

カスタムレポートテンプレートを使用すると、監視対象サーバーまたは PEM 独自の管理データベースからデータを照会する診断レポートをデザイン、実行、および管理できます。セクションに編成された SQL クエリまたはバッチ/シェルスクリプトを使用してレポートを定義でき、結果はテーブルまたはテキスト値として表示されます。

##### レポートタイプ

##### ターゲット別

カスタムレポートテンプレートは、照会する内容に基づいて、2 つのカテゴリに分類されます。

- **PEM レポート**は、PEM 独自の管理データベースを照会します。これらを使用して、PEM 監視データ、エージェントステータス、アラート履歴、および構成についてレポートします。PEM レポートでは、`pem_admin` または `pem_super_admin` ロールとターゲットサーバーを PEM エージェントバージョン 10.5 以降にバインドする必要があります。
- **非 PEM レポート**は、監視対象サーバーのデータベースを直接照会します。これらは、ワークロード分析、オブジェクトレベルの検査、またはアプリケーション固有の診断に使用します。非 PEM レポートでは、ターゲットサーバーを PEM エージェントバージョン 10.4 以降にバインドする必要があります。

##### 出身地別

- **システム定義のテンプレート**は PEM に組み込まれ、レポートテンプレートにアクセスできるすべてのユーザーが利用できます。システム定義のテンプレートは変更または削除できません。
- **カスタムテンプレート** は、`pem_manage_report_templates` 以上のロールを持つユーザーが作成および管理します。

##### 前提条件

カスタムレポートテンプレートを介してスクリプトまたは SQL クエリーを実行するには

- ターゲットサーバーは PEM エージェントバージョン 10.4 以降にバインドする必要があります。
- シェル/バッチスクリプトの場合、エージェントはホストオペレーティングシステムでコマンドを実行するために必要な権限を持っている必要があります。

##### レポートテンプレートの管理

管理インターフェイスにアクセスするには、 **Management > Reports > Manage Report Templates** に移動します。

### カスタムレポートテンプレートの作成

一意のタイトルと機能の説明で定義されるトップレベルのコンテナ。

- **Create New Report** を選択します。
- **タイトル** レポートの一意の名前を入力します必須。
- **説明** レポートの目的の説明を提供しますオプショナル。

### レポート構造の定義

カスタムレポートテンプレートは、階層構造を使用してデータを整理します。

- **レポート** トップレベルのコンテナ。 1 つ以上の **セクション**が含まれます。
- **セクション** レポート内の論理グループ。 1 つ以上の **アイテム**が含まれています。
- **Item** 個々の実行単位。

**項目定義** セクション内の各項目は、次の属性によって定義されます。

| Name | Description |
|---|---|
| Title | 特定の項目の名前。 |
| Description | 機能的なコンテキストまたは説明を提供するオプショナルフィールド。 |
| Kind | 実行エンジン **SQL Query** または **Batch/Shell Script** を指定します。 |
| Format | プレゼンテーションレイヤー **Table** 表形式出力または **Text** 単一値を定義します。 |
| Code | エージェントによって実行される特定の SQL ステートメントまたはスクリプトロジック。 |

**Format** 属性は、項目結果の表示方法を決定します。

| Format | Presentation mode | Best use case |
|---|---|---|
| Table | An interactive data grid complete with column headers. | Multi-row SQL result sets or structured log outputs. |
| Text | A single string, scalar value, or block of raw text. | Summary metrics, version strings, or simple status checks. |

### セクションタイプ

レポートの各セクションは、単純または反復です。

- **シンプルセクション**は、1 つのターゲットに対して固定された項目セットを実行します。これらは、スコープが事前にわかっている標準の診断クエリに使用します。
- **反復セクション** には、ターゲットのリストを返す **グループクエリー**が含まれます。グループクエリによって返される行ごとに、セクションの項目が 1 回実行されます。反復セクションは、データベース、スキーマ、またはオブジェクトの動的なセットにわたって同じクエリを繰り返す必要がある場合

に使用します。反復セクションでは、ターゲットサーバーを PEM エージェントバージョン 10.5 以降にバインドする必要があります。

たとえば、サーバー上のすべてのデータベースを返すグループクエリを使用した反復セクションは、データベースごとに 1 回各項目を実行します。データベース間の比較に役立ちます。

### レポートの実行

レポートを実行するには 2 つの方法があります。

**サーバーを右クリックします。**

1. PEM ブラウザツリーで、ターゲットサーバーを選択します。PEM エージェントバージョン 10.4 以降にバインドする必要があります。
2. サーバーを右クリックし、 **Run Report** を選択します。
3. ドロップダウンから目的のレポートテンプレートを選択します。
4. 選択したレポートについて、ターゲットデータベースを選択します。
5. **Run** を選択します。PEM はエージェントジョブをトリガーしてレポートを実行します。

**スケジュールされたジョブを使用する**

1. PEM ブラウザツリーで、エージェントノードを展開し、 **Jobs** を右クリックして **Create Job** を選択します。
2. ジョブダイアログで、 **Steps** タブに移動し、 **+** を選択してステップを追加します。
3. **Type** を **Report** に設定し、レポートテンプレート、ターゲットサーバー、およびデータベースを選択します。
4. スケジュールを構成し、 **Save** を選択します。

非 PEM レポートを生成するには、`pem_manage_report_templates` ロール以上が必要です。 PEM レポートの生成には、`pem_admin` または `pem_super_admin` が必要です。

> **注釈**
>
> 非 PEM レポートは PEM データベースで実行できません。 PEM 自分自身の管理データを照会するには、代わりに PEM レポートを作成します *pem_admin* または `pem_super_admin` ロールが必要です。

### 実行履歴の表示

トリガーされたレポートと監査履歴データのステータスをモニタするには、**Management > Reports > Report Execution History** に移動します。

実行履歴は、すべてのカスタムレポートテンプレートアクティビティの一元的な監査証跡を提供します。特定のレコードを見つけるには、次のフィルターを適用します。

- **サーバー名**レポートが実行された特定のターゲットサーバーでフィルタリングします。
- **エージェント名**実行の処理を担当する PEM エージェントによるフィルター。

- **ステータス** 実行状態によるフィルタリングたとえば、 **Success** 、**Partially Successful** 、 **Failed** 、 **In Progress** 、または **Unknown** など。
- **時間範囲**特定の時間ウィンドウを定義して、最近または履歴データを分離します。

### レポート結果の表示

レポートの詳細な出力にアクセスするには、 **レポートの実行履歴**ダッシュボード内の実行レコードを選択します。

**結果の構成**結果は、元のテンプレートの階層に従って構造化されています。データは**セクション** に分割され、各セクションにはそれぞれの **アイテム**が含まれます。

**データプレゼンテーションレイヤー**結果は、レポートテンプレートで設定された **Format** 属性に基づいて、インタラクティブな **Table** グリッドまたは **Text** スカラー値としてレンダリングされます。

**エラーの処理**レポート内のアイテムの実行に失敗した場合、システムは SQL エンジンまたはシェル環境から返された特定のエラーメッセージをキャプチャして項目の結果ブロック内に直接表示します。これにより、**部分成功**レポートでも、個々のポイント障害を特定してトラブルシューティングを行うことができます。

**結果のエクスポート**外部レポートまたはさらなるデータ処理のために、次の形式で実行されたレポートをダウンロードできます。

- **HTML:** 簡単な共有とブラウザベースのレビューのためのフォーマットされたドキュメント。
- **JSON** 自動ワークフローまたはサードパーティツールとの統合に最適化された構造化データファイル。

> **注釈**
>
> PEM レポート結果は、実行したユーザーのプライベートです。他のユーザーはそれらを表示することはできません。 *pem_super_admin* は、すべてのユーザーの PEM レポート結果を表示できます。

### レポートのインポートとエクスポート

PEM では、標準化された JSON ファイルを介してレポートテンプレートの移行とバックアップができます。

**レポートテンプレートのエクスポート**レポート構成をバックアップまたは共有するには

- **Management > Reports > Manage Report Templates** に移動します。
- リストから 1 つ以上のテンプレートを選択します。
- **Export** を選択して、選択した構成を JSON ファイルとしてダウンロードします。

**レポートテンプレートのインポート**テンプレートを新しいまたは既存の PEM インスタンスに一括追加するには

- **Manage Report Templates** インターフェイスを開きます。
- **Import** ボタンを選択します。
- レポートテンプレート定義を含む有効な JSON ファイルをアップロードします。

> **注釈**
>
> 同じタイトルのテンプレートが既に存在する場合、エントリをスキップするか上書きするように求められます。

### ステータス定義の理解

[ **レポート実行履歴**] タブは、各レポートジョブのライフサイクルを追跡します。次の表は、カスタムレポートテンプレートの考えられる状態を定義しています。

| Status | Definition |
|---|---|
| Success | Every item within the report template executed successfully. |
| Partially Successful | At least one item executed successfully, while others encountered errors. |
| Failed | Every item in the report failed to execute or return data. |
| In Progress | The report is currently being processed by the target PEM agent. |
| Unknown | The agent job was aborted, timed out, or the connection was lost. |

### SQL 行数の制限

パフォーマンスを最適化し、PEM サーバーでの過剰なメモリ消費を防ぐため、カスタムレポートテンプレートは、SQL クエリによって返される行数の制限を適用します。

この制限は、 **サーバー構成**ダッシュボードを介してグローバルに管理され、管理者は大規模なデータセットに対応するように値を変更できます。

| Parameter | Default Value | Description |
|---|---|---|
| *report_sql_row_limit* | 500 | レポート内の単一の SQL クエリ項目によって返される行の最大数を制限します。 |

> **注釈**
>
> この制限を増やすと、大量のデータを処理する場合のレポート生成速度と**レポート結果**ビューの全体的な応答性に大きな影響を与える可能性があります。

### ロールと権限

PEM は 4 つのロールを使用して、レポートテンプレートと結果へのアクセスを制御します。

``**pem_view_reports**`` — レポート実行履歴への読み取り専用アクセス。

- システム定義およびカスタムの非 PEM レポートを表示します。

レポートの生成、レポートテンプレートの作成、編集、表示、または削除はできません。

``**pem_manage_report_templates**`` — すべての `pem_view_reports` 権限に加えて、以下が含まれます。

- カスタムの非 PEM レポートテンプレートを作成および編集します。
- 非 PEM レポートを生成します。
- 独自の非 PEM レポート結果を削除します。

PEM レポートテンプレートを作成または編集したり、PEM レポートのレポート実行履歴を表示したりすることはできません。

``**pem_admin**`` — すべての `pem_manage_report_templates` 権限に加えて、以下が含まれます。

- カスタム PEM レポートテンプレートを作成および編集します。
- PEM レポートを生成します。
- 独自の PEM レポートの実行履歴を表示します。

他のユーザーの PEM レポート結果は表示できません。

``**pem_super_admin**`` — すべての `pem_admin` 権限に加えて、以下が含まれます。

- すべてのユーザーの PEM レポート結果を表示します。
- ユーザーのレポート結果を削除します。

システム定義のテンプレートは、ユーザーまたはロールが変更または削除することはできません。

### アクセス許可のクイックリファレンス

#### テンプレート

| Role | View | Create/Edit | Delete |
|---|---|---|---|
| *pem_view_reports* | | | |
| *pem_manage_report_tempi* | non-PEM | non-PEM | non-PEM |
| *pem_admin* | all | all | all |
| *pem_super_admin* | all | all | all |

#### レポート

| Role | View | Generate | Delete |
|---|---|---|---|
| *pem_view_reports* | non-PEM | | |

次のページに続く

表 53 – 前のページからの続き

| Role | View | Generate | Delete |
|---|---|---|---|
| *pem_manage_report_templ* | non-PEM | non-PEM | own non-PEM |
| *pem_admin* | non-PEM + own PEM | all | own non-PEM + own PEM |
| *pem_super_admin* | all | all | all |

#### 8.12.2.2 Viewing system reports

PEM 10.5 では、**管理>レポート**にアラート履歴レポートのみが表示されます。システム構成レポートとコア使用状況レポートは、**管理>レポート>管理レポートテンプレート**の下でシステム定義のテンプレートとして利用できるようになりました。

`superuser` または `pem_admin` ロールを持つユーザーのみがシステム構成またはコア使用率レポートをダウンロードできます。

これらのレポートの情報は、最新のプローブランタイムを示しています。

##### システム定義のレポートテンプレート

PEM では、レポートを完全にカスタマイズできますが、データベースの詳細な分析用に設計された事前構成されたシステムレポートテンプレートも含まれています。これらのレポートはプラットフォームに組み込まれており、手動テンプレート構成を必要とせずにすぐに値を提供します。

##### サーバーダウンタイムレポート

サーバーダウンタイムレポートは、監視対象の PostgreSQL サーバーからのハートビートギャップを追跡することにより、サーバーの可用性とダウンタイムの分析を提供します。計画的なダウンタイムブラックアウト期間と計画外のダウンタイムを区別し、過去 30 日間の各サーバーの可用性の割合を計算します。

**可用性**は、計画外のダウンタイムのみ `1 - (Unplanned Downtime / Total Time)` から計算されます。ブラックアウト期間中の計画的なダウンタイムは、可用性の計算に影響を与えません。

レポートは、PEM 10.5 にアップグレードすると、自動的にデータの収集を開始します。手動設定は必要なく、無効にすることはできません。

> **注釈**
>
> このレポートを実行するには、ターゲットサーバーのエージェントサーバーバインディング ASB ユーザーに `pem_user` ロールが必要です。これは最も一般的なセットアップの問題です。以下の **オブジェクト権限** を参照してください。

##### レポートへのアクセス

サーバーダウンタイムレポートを実行するには、次のいずれかの方法を使用します。

**実行レポートを使用する**

1. ブラウザツリーで、サーバーを選択します。

2. 右クリックして [ **Run Reports** ] を選択します。
3. **Report** ドロップダウンで、 **Server Downtime Report** を選択します。
4. データベースドロップダウンで、データベースを選択します。
5. **Run** を選択します。

#### スケジュールされたジョブを使用する

1. オブジェクトツリーで、エージェントを選択して右クリックし、 **Create Job** を選択します。
2. ジョブ名を指定し、 **Steps** タブに移動します。
3. ステップ名を指定し、 **Kind** を **Report** に設定し、レポート、サーバー、およびデータベースを選択します。
4. オプションでスケジュールを構成し、ジョブを保存します。
5. ジョブを右クリックし、 **Run Now** を選択します。

レポート出力を表示またはエクスポートするには、 **Management > Reports > Report Execution History** に移動します。

### レポートセクション

レポートには 3 つのセクションが含まれています。

#### 概要メトリック

すべての監視対象サーバーの概要を提供します。

- **監視対象サーバーの合計** - 過去 30 日間のダウンタイムデータを持つ一意のサーバーの数。
- **平均可用性の割合** - すべての監視対象サーバーの平均可用性。
- **合計ダウンタイム時間** - すべてのサーバーにわたるすべてのダウンタイム計画および計画外の合計。
- **計画外のダウンタイムの合計時間** - ブラックアウト期間以外で発生したダウンタイムの合計。
- **計画的なダウンタイムの合計時間** - ブラックアウト期間に発生したダウンタイムの合計。

#### サーバーの可用性の詳細

サーバーごとの内訳を表示し、可用性の割合の昇順で並べ替えますワーストパフォーマーを最初に表示します。

- **サーバー名** - 監視対象サーバーのホスト名または IP アドレス。
- **サーバーの説明** - ユーザー定義の説明。
- **可用性 pct** - `1 - (Unplanned Downtime / Total Time)` として計算されます。
- **合計時間** - 合計ダウンタイム計画および計画外。
- **計画外の時間** - ブラックアウト期間以外のダウンタイム。
- **計画時間** - 停電期間中のダウンタイム。
- **データベースカウント** - サーバー上の非システムデータベースの数。

### ワーストパフォーマートップ 10

過去 30 日間で可用性が最も低い 10 サーバーを表示します。サーバー可用性の詳細セクションと同じ列を使用します。このセクションを使用して、注意が必要なサーバーをすばやく特定します。

#### 構成パラメーター

pem_admin ロールを持つユーザーは、 **サーバー構成**でこれらのパラメーターを調整できます。

| Parameter | Default | Description |
|---|---|---|
| *downtime_retention_days* | 30 | 保持するダウンタイム履歴の日数。古いデータは 1 日 2 回自動的にパージされます。この値を減らすと、ストレージの使用量は削減されますが、履歴傾向分析が制限されます。 |
| *downtime_heartbeat_gap_sec* | 60 | PEM がダウンタイムイベントを記録するまでのハートビート間の最小ギャップ秒単位。この値を増やして、高遅延のネットワーク環境での誤検出を減らします。 |

#### アクセス許可

レポートは、各ターゲットサーバーの ASB ユーザーを使用して実行されます。レポート機能を実行するには、そのユーザーに pem_user ロールが必要です。

ロールを付与するには、スーパーユーザーとして PEM データベースに接続し、実行します。

```
GRANT pem_user TO <asb_username>;
```

#### 制限事項

- **ダウンタイムの原因は不明です** - レポートでは、サーバーのダウン、PEM エージェントのダウン、ネットワークパーティション、またはファイアウォールの問題を区別できません。しきい値を超えるすべてのハートビートギャップは、サーバーのダウンタイムとして記録されます。計画的なメンテナンスにブラックアウト期間を使用して、このデータの意味を維持します。
- **短期間の停止はキャプチャされません** - 最小検出可能なダウンタイムは、ハートビートギャップしきい値デフォルト 60 秒とほぼ同じです。この期間よりも短い停止はレポートに表示されません。
- **アップグレード前の履歴なし** - このレポートは、PEM 10.5 以降へのアップグレード時点からのダウンタイムのみを追跡します。以前の PEM バージョンの履歴データはありません。完全なレポートウィンドウには、アップグレード後最大 30 日かかります。
- **高遅延ネットワーク** - 高遅延環境では、通常のハートビート遅延が 60 秒の検出しきい値を超え、誤ったダウンタイムイベントを生成する場合があります。正常なサーバーで予期しないダウンタイムが発生した場合は、downtime_heartbeat_gap_seconds を 90〜120 秒に増加します。
- **ブラックアウトデータの依存関係** - 計画的なダウンタイムの分類には、PEM にブラックアウト期間が存在し、「プロセスアラートブラックアウト」ジョブが有効になって実行されている必要がありま

す。ブラックアウトデータが利用できない場合、すべてのダウンタイムは計画外として分類され、可用性の割合は実際より低くなります。

- **30 日のデフォルトウィンドウ** - デフォルトでは、過去 30 日のデータのみを利用できます。`downtime_retention_days` を増やすとデータの保持期間が延長されますが、レポートには常に最新の 30 日のみが表示されます。
- **ハードコードされた 30 日のサーバー可用性ウィンドウ** - レポートは、過去 30 日間のサーバー可用性データのみを取得するようにハードコードされています。より長いレポート期間が必要な場合は、カスタムレポートテンプレートを作成します。

### EDB Postgres ワークロードレポート PWR

EDB Postgres ワークロードレポート PWR は、特定のスナップショット期間におけるサーバーのパフォーマンスとリソース使用率の包括的な概要を提供します。これは、ボトルネック、高負荷の SQL ステートメント、および待機イベントを特定するために重要なツールです。

> **注釈**
>
> このレポートテンプレートを実行するには、 *edb_wait_states* 拡張機能がターゲットデータベースにインストールされ有効になっていることを確認します。

#### サーバー情報

サーバーバージョンに関する情報を表示します。

- **サーバーバージョン** - Postgres サーバーのバージョン番号。
- **アーキテクチャ** - サーバーが構築されるハードウェアアーキテクチャ。
- **システム識別子** - データベースクラスターの一意の識別子。
- **Redwood モード** - クラスターが Redwood Oracle 互換性モードで作成されたかどうかを示します。
- **現在のユーザー** - レポートを生成したユーザー。
- **実際の開始/終了スナップタイムスタンプ** - データ収集の開始と終了をマークする正確なタイムスタンプ。
- **スナップショットの継続時間** - スナップショット中に経過した合計時間。

レポートのタイムスタンプに関する情報を表示します。

- **Session counts** - スナップショットの開始と終了の両方で接続されたセッションの数。
- **データベース時間** - すべてのセッションがデータベースでの作業または待機に費やした非アイドル時間の合計秒単位。
- **Uptime** - 最後にサーバーを再起動してから費やされた合計時間。
- **Role** - サーバーがプライマリインスタンスかスタンバイインスタンスかを示します。

### ロードプロファイル

トランザクションごとの平均負荷プロファイルを示します。

- **Blocks fetched** - トランザクションごとにアクセスされたブロックの平均数。
- **WAL records/bytes** - トランザクションごとに生成された先行書き込みログレコードとバイトの平均数。
- **Transactions** - スナップショット中に処理されたトランザクションの合計数。
- **Abort rate** - アボートが発生したトランザクションの割合。

### トップの待機イベント

トップのイベントで費やされた合計待機時間秒単位で表示されます。

- **待機イベント** - 特定の待機イベントの名前。( **CPU** は、アクティブな作業に費やされた時間を示します)。
- **Wait class** - 待機イベントのカテゴリまたはタイプ。
- **待機時間** - このイベントに費やされた秒単位の合計時間。
- **Pct データベース時間** - このイベントタイプの待機に費やされた合計 `Dbtime` の割合。

### 上位の SQL ステートメント

`dbtime` でソートされた上位 10 の SQL ステートメントを表示します。

- **データベース時間** - SQL ステートメントの実行中に消費された累積データベース時間秒単位。
- **待機時間** - SQL ステートメントがシステムリソースまたはバックグラウンドイベントの待機に費やした合計時間。
- **CPU 時間** - SQL ステートメントによって使用された CPU 処理時間の合計量。
- **最上位の待機イベント** - ステートメントが最長の遅延を発生した特定の待機イベント識別子。
- **Query** - 実行されたクエリの実際の SQL テキスト。

### トランザクション統計

スナップショット期間中のトランザクション統計を表示します。

- **Xact commit** - 正常に完了および保存されたトランザクションの合計数。
- **Xact rollback** - 取り消されたまたは中止されたトランザクションの合計数。

### WAL 統計

スナップショット期間中の WAL レコードの変更に関する統計を表示します。

- **Wal records** - スナップショット中に生成された個々の WAL レコードの合計数。
- **Wal fpi** - 作成されたフルページ画像 FPI の数。これらは、チェックポイントに続いてページが最初に変更された後に発生し、WAL ボリュームが大幅に増加する可能性があります。

- **Wal bytes** - 生成されたすべての WAL データの合計サイズ、バイト単位で測定されます。

#### 共有バッファーの統計

スナップショット期間中の共有バッファの統計を表示します。

- **Blocks read** - 読み取られたディスクブロックの合計数。
- **Blocks hit** - ヒットしたブロックの合計数。

#### タプルの統計

スナップショット期間中のタプルに関する統計を表示します。

- **返されたタプル** - シークエンシャル スキャンによって返された行数。
- **Tuples fetched** - 特にインデックススキャンを介して取得された行の数。
- **挿入されたタプル** - データベースに追加された新しい行の合計数。
- **タプルが更新されました** - 変更された既存の行の合計数。
- **削除されたタプル** - データベースから削除された行の合計数。

#### 一時ファイルの統計

一時ファイルの作成に関する統計を表示します。

- **Temp files** - 作成された一時ファイルの数。
- **Temp bytes** - 一時ファイルに書き込まれたデータの合計ボリューム。

#### ユーザーセッション情報

スナップショット期間中に接続/実行されているセッション情報

- **セッション ID** - `edb_wait_states` によって特定の Postgres セッションに割り当てられた一意の ID。
- **データベース名** - セッションが現在接続しているデータベースの名前。
- **Username** - セッションにログインしたデータベースユーザーの ID。
- **アプリケーション名** - このセッションに接続しているアプリケーションまたはクライアントツールの名前 `psql` 、`pgAdmin` 、またはカスタムアプリサービスなど。
- **Client hostname** - 接続されたクライアントのネットワークホスト名、接続の起点を提供します使用可能な場合。
- **セッション開始タイムスタンプ** - セッションが最初に確立されたときの正確な日時。

#### システム情報

サーバーマシンの CPU とメモリ情報を表示します。

- **ホスト名** - Postgres が実行されているマシンのネットワーク名。
- **CPU 情報** - プロセッサーの数、ベンダー、クロック速度を含む詳細。

- **メモリ情報** - ホストの合計メモリ情報。

### Postgres データベース設定

Postgres データベースの設定を示します。

- **Category** - パラメーターの論理機能グループ。
- **パラメーター** - カテゴリ内の設定の名前と現在の値。強調表示されたエントリは、カスタム値を含むパラメーターを示します。
- **Value** - 指定されたパラメーターの現在の操作設定。
- **Is custom value** - パラメーターがデフォルト値から変更されたかどうかを示すフラグ。

### コア使用率レポート

コア使用率レポートは、すべての監視対象データベースサーバー、エージェント、およびグループにわたる CPU コア割り当ての包括的なビューを提供します。キャパシティプランニング、ライセンスコンプライアンス追跡、およびインフラストラクチャ監査に使用します。`superuser` または `pem_admin` ロールを持つユーザーのみがこのレポートを実行できます。

> **注釈**
>
> PEM 10.5 では、このレポートの名前が**PEM コア使用レポート**から**コア使用レポート**に変更され、[レポートテンプレートの管理] に移動されました。既存のインストールは、アップグレード中に自動的に移行されます。手動アクションは必要ありません。

### レポートへのアクセス

コア使用率レポートを実行するには、次のいずれかの方法を使用します。

#### 実行レポートを使用する

1. ブラウザツリーで、サーバーを選択します。
2. 右クリックして [ **Run Reports** ] を選択します。
3. **レポート** ドロップダウンで、 **コア使用状況レポート**を選択します。
4. データベースドロップダウンで、PEM データベースを選択します。
5. **Run** を選択します。

#### スケジュールされたジョブを使用する

1. オブジェクトツリーで、エージェントを選択して右クリックし、 **Create Job** を選択します。
2. ジョブ名を指定し、 **Steps** タブに移動します。
3. ステップ名を指定し、 **Kind** を **Report** に設定し、レポート、サーバー、およびデータベースを選択します。
4. オプションでスケジュールを構成し、ジョブを保存します。

5. ジョブを右クリックし、 **Run Now** を選択します。

レポート出力を表示またはエクスポートするには、 **Management > Reports > Report Execution History** に移動します。

### レポートセクション

レポートには 2 つのセクションが含まれています。

#### コアサマリー

監視対象インフラ全体の高レベルの統計を提供します。

- **合計 CPU コア** - ローカルに管理されているサーバー全体のすべての CPU コアの集計数。リモート管理サーバーは、エージェントデータを利用できないため、除外します。
- **サーバータイプ別コア** - データベースエンジンごとに内訳されたコア分布 EDB Postgres Advanced Server、 EDB Postgres Extended 2ndQPG、および PostgreSQL。
- **データベースバージョン別のコア** - 特定のデータベースバージョンごとにグループ化されたコア数たとえば、「PostgreSQL 16」、「EPAS 15」、または「2ndQ PostgreSQL 14」。
- **プラットフォーム別のコア** - Linux、Windows、macOS、およびその他の Unix バリアントを含むオペレーティングシステム間のディストリビューション。プラットフォームごとのサーバー数とコア数を表示します。
- **グループ別のコア** - 組織グループ別のコア割り当て。ユーザー固有のグループ名を尊重して、非表示グループをオーバーライドおよび除外します。

#### サーバーコアの概要

管理タイプごとに編成されたサーバーレベルの詳細情報

- **ローカル管理サーバー** - PEM エージェントにバインドされたサーバー。サーバー名、ホストとポート、サーバータイプ、PGD ステータスとバージョン、プラットフォーム、CPU コア、合計 RAM MB、およびメモリ使用率を表示します。
- **リモート管理サーバー** - エージェントを使用せずにリモートで監視されるサーバー。サーバー名、ホストとポート、サーバータイプ、および PGD のステータスとバージョンを表示します。これらのサーバーは、CPU とメモリデータを利用できません。
- **非管理対象サーバー** - PEM に登録されていますが、エージェントにバインドされていないサーバー。サーバー名、ホストとポート、および PGD のステータスとバージョンを表示します。

### システム構成レポート

システム構成レポートは、ブラウザツリーの下にリストされている PEM エージェントグループ、PEM サーバーディレクトリグループ、およびカスタムグループに関する詳細情報を提供します。これらのグループには、PEM サーバー、PEM エージェント、およびデータベースサーバーを含めることができます。`superuser` または `pem_admin` ロールを持つユーザーのみがこのレポートを実行できます。

システム構成レポートを実行するには、次のいずれかの方法を使用します。

#### 実行レポートを使用する

1. ブラウザツリーで、PEM Server ノードを選択します。
2. 右クリックして [ **Run Reports** ] を選択します。
3. **Report** ドロップダウンで、 **System Configuration Report** を選択します。
4. データベースドロップダウンで、PEM データベースを選択します。
5. **Run** を選択します。

#### スケジュールされたジョブを使用する

1. オブジェクトツリーで、エージェントを選択して右クリックし、 **Create Job** を選択します。
2. ジョブ名を指定し、 **Steps** タブに移動します。
3. ステップ名を指定し、 **Kind** を **Report** に設定し、レポート、サーバー、およびデータベースを選択します。
4. オプションでスケジュールを構成し、ジョブを保存します。
5. ジョブを右クリックし、 **Run Now** を選択します。

出力を表示またはダウンロードするには、 **Management > Reports > Report Execution History** に移動し、リストでレポートを見つけて、 **Download Report** 列で **HTML** または **JSON** を選択します。

### レポートセクション

レポートには 7 つのセクションが含まれています。

#### Postgres Enterprise Manager の概要

PEM バックエンドに関する詳細を提供します。

- Postgres Enterprise Manager バックエンドデータベースサーバーバージョン
- アプリケーションバージョン
- アプリケーションにアクセスするユーザー名
- プラットフォーム固有の情報

#### 概要

システムに登録されている PEM エージェントと監視対象サーバーの合計数を提供します。

#### モニタリングの概要

監視ワークロードのリアルタイムのスナップショットを提供します。

- 有効になったアラートの数
- 1 時間ごとのアラート評価
- 1 時間あたりのプローブ実行数
- 過去 24 時間のアクティブなエージェント
- 過去 24 時間のアクティブサーバー

#### アラート評価情報

アラート処理パフォーマンスに関する統計を提供します。

- アラート処理時間メトリック最大、最小、平均、p50、p90、p95、および p99
- アクティブなエージェントアラート設定 —SMTP、SNMP、および Webhook 通知が有効になっているエージェントの数
- 遅延アラート —スケジュールされた評価時間より遅れて実行されている上位 50 のアラート

#### サーバー情報

各監視対象データベースサーバとそのオブジェクト数に関する詳細を提供します。

- データベースサーバーバージョン
- ホスト
- ポート
- データベース名
- データベースのサイズ
- テーブル領域のサイズ
- オブジェクト数データベース、スキーマ、テーブル、インデックス、ビュー、ファンクション、拡張機能、外部キー、シーケンス、テーブルスペース
- スキーマごとのテーブルインデックスの概要

グループサーバー名は、サーバーが追加されるグループ名によって異なります。

#### エージェント情報

PEM エージェント、CPU コア、ディスク使用率、およびメモリ情報の詳細を提供します。

#### サイズ情報

キャパシティプランニングをサポートするデータを提供します。

- オブジェクトカタログの概要 - エージェント、サーバー、データベース、スキーマ、テーブル、インデックスの数
- `pem` 、`pemdata` 、および `pemhistory` スキーマの PEM 内部テーブルサイズ

### アラート履歴レポート

Alert History レポートは、エージェントまたはサーバーレベルでのアラート履歴に関する詳細情報を JSON 形式で提供します。

**管理>レポート>アラート履歴レポート**を選択します。ダイアログボックスから、オプションを選択します。

- **Agent/Server Name** - リストからエージェントとサーバーを選択します。**Overall System Report** オプションは、すべての登録済みおよびアクティブなエージェントとサーバーのレポートを生成するために使用できます。
- **タイムフレーム** - リストから必要なタイムフレームを選択します。

- **Alert Types** - アラート履歴レポートを生成するアラートタイプを選択します。

## 8.13 PEM command line interface

PEM CLI を使用すると、アラートテンプレートまたはプローブを PEM サーバーから別の PEM サーバーにエクスポートおよびインポートできます。

edb.repo リポジトリとは別にパッケージ edb-pem-cli-8.5.0-1-<platform_name> を使用して PEM CLI をインストールできます。バイナリファイル/usr/edb/pem/cli/bin/pem とライセンスファイル/usr/edb/pem/edb-pem-cli-license.txt を作成します。

PEM CLI パッケージをインストールした後、./pem -h コマンドを使用して、使用可能なすべてのオプションを表示します。

```
$ ./pem -h
__OUTPUT__
PEM CLI (Postgres Enterprise Manager Command Line Interface)
Usage: ./pem [OPTIONS] [SUBCOMMAND]

Options:
  -h,--help                   Print this help message and exit
  -v,--version                Show the app version and exit

Logging:
  --log-file [/home/asheshvashi/.pem-cli.log]
                              Set the log file

                              NOTE: Directory must exist

  -l,--log-level :value in {FATAL->1,ERROR->2,WARNING->3,DEBUG->5,INFO->4,VERBOSE->6}␣
↪OR {1,2,3,5,4,6} [INFO]
                              Set the log level

PEM Information:
  -p,--pem-url REQUIRED (Env:PEM_SERVER_URL)
                              Set the URL for accessing Postgres Enterprise Manager
                              e.g.
                              https://127.0.0.1:8443/pem
                              https:://pem_host:8443/pem

  -u,--pem-user REQUIRED (Env:PEM_API_USER)
                              User to access the PEM REST API

  -f,--password-file (Env:PEM_PASSWORD_FILE)
                              Location of the file, which contains the password.
                              The permission on a password file must disallow any␣
↪access to world or group.
```

(次のページに続く)

(前のページからの続き)

```
                              NOTE:
  - The first non-empty line in the file will be treated as a
                                password.
  - Environment variable PEM_API_PASSWORD can be used for
                                providing the password instead of the password file

  -i,--insecure               Set the insecure connection


Subcommands:
  import-alert-templates      Import the alert template(s) into the PEM server.
  list-alert-templates        List the alert templates in the PEM server.
  export-alert-templates      Export the custom (user-defined) alert templates from␣
↪the PEM server.
  import-probes               Import the probe(s) into the PEM server.
  export-probes               Export the custom (user-defined) probes from the PEM␣
↪server.
  list-probes                 List the probes in the PEM server.
```

import-alert-templates サブコマンドで使用可能なすべてのオプションを表示できます。

```
$ ./pem import-alert-templates -h
__OUTPUT__
Import the alert template(s) into the PEM server.
Usage: ./pem import-alert-templates [OPTIONS]

Options:
  -h,--help                   Print this help message and exit
  -i,--in-file                Alert template file
  --ignore-existing           Ignore the existing alert template in the PEM server
  --ignore-existing-probes    Ignore the existing probes in the PEM server
```

list-alert-templates サブコマンドで使用可能なすべてのオプションを表示できます。

```
$ ./pem list-alert-templates -h
__OUTPUT__
List the alert templates in the PEM server.
Usage: ./pem list-alert-templates [OPTIONS]

Options:
  -h,--help                   Print this help message and exit
  -c,--alert-template-category :value in {ALL->2,CUSTOM->1,SYSTEM->0} OR {2,1,0}␣
↪[CUSTOM]
                              List down the alert templates of this category
```

export-alert-templates サブコマンドで使用可能なすべてのオプションを表示できます。

```
$ ./pem export-alert-templates -h
__OUTPUT__
Export the custom (user-defined) alert templates from the PEM server
Usage: ./pem export-alert-templates [OPTIONS]

Options:
  -h,--help                    Print this help message and exit
  -i,--id INT ...              List of id for the custom alert templates to be exported
  -o,--out-file [Standard output (console)]
                               Output file name
  -w,--overwrite               Overwrite the existing output file
```

import-probes サブコマンドで使用可能なすべてのオプションを表示できます。

```
$ ./pem import-probes -h
__OUTPUT__
Import the probe(s) into the PEM server.
Usage: ./pem import-probes [OPTIONS]

Options:
  -h,--help                    Print this help message and exit
  -i,--in-file                 Probes file
  --ignore-existing            Ignore the existing custom probes in the PEM server
```

export-probes サブコマンドで使用可能なすべてのオプションを表示できます。

```
$ ./pem export-probes -h
__OUTPUT__
Export the custom (user-defined) probes from the PEM server
Usage: ./pem export-probes [OPTIONS]

Options:
  -h,--help                    Print this help message and exit
  -i,--probe-id INT ...        List of the custom probe-ids to be exported
  -o,--out-file [Standard output (console)]
                               Output file name
  -w,--overwrite               Overwrite the existing output file
```

list-probes サブコマンドで使用可能なすべてのオプションを表示できます。

```
$ ./pem list-probes -h
__OUTPUT__
List the probes in the PEM server.
Usage: ./pem list-probes [OPTIONS]

```

(次のページに続く)

(前のページからの続き)

```
Options:
  -h,--help                    Print this help message and exit
  -c,--probe-category :value in {ALL->2,CUSTOM->1,SYSTEM->0} OR {2,1,0} [CUSTOM]
                               List down the probes of this category
```

### 8.13.1 例

すべてのプローブシステムおよびカスタムプローブをリストするには、-c all オプションを使用します。

```
$ PEM_PASSWORD_FILE=/tmp/pass PEM_API_USER=postgres PEM_SERVER_URL=https://172.17.0.4/
↪pem ./pem -i list-probes -c all
```

そこで

- PEM_PASSWORD_FILE=/tmp/pass は、postgres ユーザーのパスワードを含むパスワードファイルです。
- PEM_API_USER=postgres は、PEM CLI を実行しているユーザー名です。
- PEM_SERVER_URL=https://172.17.0.4/pem は、プローブを含む PEM サーバーの URL です。
- -i list-probes は、 list-probes サブコマンドを使用した insecure オプションであり、安全でない接続を許可し、プローブをリストします。
- -c all は、すべてのシステムとカスタムプローブをリストするための all 値を備えた-c オプションです。

## 8.14 EDB Postgres Enterprise Manager REST APIs v1.0.0

**EDB Postgres Enterprise Manager** は、トークンベースの認証メカニズムを使用して、公開された REST API へのアクセスを許可します。

PEM は認証のためにトークンを使用します。トークンを生成するには、トークンセクションを参照してください。トークン文字列を受け取った後、https://PEM-SERVER-IP/api/token/以外のリソースを呼び出すときに、’X-Auth-Token’ヘッダーにトークン文字列を含める必要があります。

**注**

- **EDB Postgres Enterprise Manager このドキュメントでは PEM と呼ばれています。**
- **PEM** *自己認証 SSL 証明書を使用するため、「SSL 証明書の問題: 自己署名証明書」などのエラーが発生する場合があります。’非セキュア’接続を使用してサーバーに接続する必要がある場合があります。

**残りの API バージョン**

- v17 現在

ベース URL

- https://PEM-SERVER-IP/pem/api

電子メール EnterpriseDB Web EnterpriseDB ライセンス限定使用ソフトウェアライセンス契約

### 8.14.1 トークン

認証のために PEM REST API によって生成および使用される文字列

#### 8.14.1.1 トークン生成

**コードサンプル**

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/token/ \
  -H Content-Type: application/json \
  -H Accept: application/json
```

```
import requests
headers = {
  Content-Type: application/json,
  Accept: application/json
}

r = requests.post(https://PEM-SERVER-IP/pem/api/token/, headers = headers)

print(r.json())
```

```
const inputBody = {
  "username": "string",
  "password": "string"
};
const headers = {
  Content-Type:application/json,
  Accept:application/json
};

fetch(https://PEM-SERVER-IP/pem/api/token/,
{
  method: POST,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

```
POST /token/
```

**認証用のトークンを生成します。**

この URL は、PEM が公開する他の REST API が使用するトークンを作成します。有効なユーザー名とパスワードを入力する必要があります。プロセス中に検証されます。

トークンの有効期間は 20 分です。

**ボディパラメーター**

```
{
  "username": "string",
  "password": "string"
}
```

#### 8.14.1.2 パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| body | body | object | true | 最初に/token/ URL を使用してトークンを作成します。 |
| » username | body | string | true | pem_user ロールを持つユーザー名を検証します。 |
| » password | body | string | true | ユーザー名のパスワード |

**応答例**

```
{
  "issuedAt": "2019-08-24T14:15:22Z",
  "expiresAt": "2019-08-24T14:15:22Z",
  "schema_version": 0,
  "backend_schema_version": 0,
  "version_long": 0,
  "version": "string",
  "supported_api_versions": ["string"],
  "deprecated_api_versions": ["string"]
}
```

#### 8.14.1.3 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 201 | *8.* スケジュールされた停電が作成されましたダイアログ | トークンが作成され、応答ヘッダーに'X-Subject-Token' として見つかります。 | Inline |
| 401 | Unauthorized | 指定された情報は有効ではありません。 | None |
| 500 | Internal Server Error | 何かがうまくいかなかった。 | None |
| 503 | Service Unavailable | PEM バックエンドデータベースサーバーがダウンしています。現在、サービスはご利用いただけません。 | None |

#### 8.14.1.4 レスポンススキーマ

ステータスコード **201**

_トークンの生成日と有効期限がいつあるかに関連する情報が含まれています。

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » issuedAt | string(date-time) | false | none | この時点でトークンが生成されました。 |
| » expiresAt | string(date-time) | false | none | トークンの有効期限が切れる日時を提供します。 |
| » schema_version | integer | false | none | アプリケーションがサポートするスキーマバージョン |
| » backend_schema_version | integer | false | none | バックエンドデータベーススキーマバージョン |
| » version_long | integer | false | none | ロングフォーマットで表されるバージョン |
| » version | string | false | none | PEM バージョンの文字列表現 |
| » supported_api_versions | [string] | false | none | サポートされている API バージョンのリスト |
| » deprecated_api_versions | [string] | false | none | 非推奨の API バージョンのリスト |

#### 8.14.1.5 レスポンスヘッダー

| Status | Header | Type | Format | Description |
|---|---|---|---|---|
| 201 | X-Subject-Token | string | | 新しいトークンを返します。これは、さらなる操作のためにREST API で使用できます |

> **注釈**
>
> この操作には認証が必要ありません

#### 8.14.1.6 delete_*token*

**コードサンプル**

```
#  You can also use wget

curl -X DELETE https://PEM-SERVER-IP/pem/api/token/ \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  X-Auth-Token: string
}

r = requests.delete(https://PEM-SERVER-IP/pem/api/token/, headers = headers)

print(r.json())
```

```
const headers = {
  "X-Auth-Token": "string",
};

fetch("https://PEM-SERVER-IP/pem/api/token/", {
  method: "DELETE",

  headers: headers,
})
  .then(function (res) {
    return res.json();
  })
```

(次のページに続く)

(前のページからの続き)

```
  .then(function (body) {
    console.log(body);
  });
```

```
DELETE /token/
```

**トークンを解放します。**

トークンを無効にします。今後の操作では使用できなくなります。

#### 8.14.1.7 パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| X-Auth-Token | header | string | true | 承認トークン |

**応答例**

#### 8.14.1.8 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 204 | No Content | PEM からトークンを正常に解放しました。 | |
| This token can not be used for any further operations. | None | NA | |

#### 8.14.1.9 レスポンススキーマ

**注釈**

この操作には認証が必要ありません

### EDB Postgres Enterprise Manager REST APIs v17.0

**注**

- **EDB Postgres Enterprise Manager このドキュメントでは**** PEM **と呼ばれています。*
- **PEM は自己認証 *SSL* 証明書を使用するため、「*SSL* 証明書の問題*:* 自己署名証明書」のようなエラーが発生する場合があります。’ *insecure*’ 接続を使用してサーバーに接続する必要がある場合があります。**

**PEM** は、次の操作のエンドポイント REST API を公開します。

- **監視対象サーバーおよびエージェントから取得したオブジェクトの取得。**
- プローブの ***C** reate、 **R** etrieve、 **U** pdate、および **D** elete CRUD オペレーション。
- **さまざまなプローブによって収集された統計/メトリックデータ履歴データを含むの取得。**
- アラートの ***C** reate、 **R** etrieve、 **U** pdate、および **D** elete CRUD オペレーション。
- アラートメカニズムで使用されるメールグループの ***C** reate、 **R** etrieve、 **U** pdate、および **D** elete CRUD オペレーション。
- アラートで使用されるアラートテンプレートの ***C** reate、 **R** etrieve、 **U** pdate、および **D** elete CRUD オペレーション。
- プローブ構成の **R 取得および U** *pdate*。
- **エージェント/サーバー/データベース/スキーマレベルの プローブ構成を、同じレベルまたは下位レベルのオブジェクトの他のオブジェクトにコピーします。**
- *1* **つのオブジェクトのアラート構成を他のオブジェクトにコピーします。**
- PEM サーバー構成を **R** *etrieve* **および U** *pdate* **します。**

**v17 の機能強化**

- **アラートのブラックアウトをファーストクラスオブジェクト** *CRUD* **として管理するための** `*/blackout/*` **エンドポイントを追加します。**
- – `name` および `name_contains` クエリパラメーターを介してオブジェクトリストエンドポイントの名前ベースのフィルタリングサポートを追加します。

ベース URL

- https://PEM-SERVER-IP/pem/api/v17

電子メール EnterpriseDB Web EnterpriseDB ライセンス限定使用ソフトウェアライセンス契約

### Object

オブジェクトは、pem エージェント、監視対象サーバー、またはデータベースオブジェクトつまりデータベース、スキーマ、テーブル、インデックス、シーケンス、ファンクション、またはビューです。

#### get_*server_group*

**コードサンプル**

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/server_group/ \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
```

(次のページに続く)

(前のページからの続き)

```
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/server_group/, headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch("https://PEM-SERVER-IP/pem/api/v17/server_group/", {
  method: "GET",

  headers: headers,
})
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /server_group/

**サーバーグループのリストを取得します。**

サーバーグループのリストを返します。 name 完全一致または name_contains 部分文字列検索クエリパラメーターを介してオプションの名前ベースのフィルタリングをサポートします。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| X-Auth-Token | header | string | true | 認証用のトークン |
| name | query | string | false | 完全な名前一致大文字と小文字を区別してサーバーグループをフィルタリングします。 name_contains と相互排他的。 |
| name_contains | query | string | false | 部分文字列の一致大文字と小文字を区別してサーバーグループをフィルタリングします。 name と相互排他的。 |

### 応答例

```
{
  "id": 0,
  "name": "string",
  "hidden": true,
  "parent_id": 0
}
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | サーバーグループのリストを返します。 | *ServerGroupBodyGet* |
| 400 | Bad Request | 相互排他性違反 `name`または類似の完全一致フィルタと`name_contains`または類似の部分文字列フィルタが両方とも同じ要求で指定されました。フィルタリングは、ペアのうちの 1 つのみを受け入れます。 | Inline |

### レスポンススキーマ

ステータスコード **400**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » success | integer | false | none | なし |
| » errormsg | string | false | none | なし |

**注釈**

この操作には認証が必要ありません

### post_server_group

コードサンプル

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v17/server_group/ \
  -H Content-Type: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Content-Type: application/json,
  X-Auth-Token: string
}

r = requests.post(https://PEM-SERVER-IP/pem/api/v17/server_group/, headers = headers)

print(r.json())
```

```
const inputBody = {
  "name": "string"
};
const headers = {
  Content-Type:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v17/server_group/,
{
  method: POST,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

POST /server_group/

新しいサーバーグループを追加します。

ボディパラメーター

```
{
```

(次のページに続く)

(前のページからの続き)

```
  "name": "string"
}
```

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| X-Auth-Token | header | string | true | 承認トークン |
| body | body | *Server-GroupBody* | true | なし |

### 応答例

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ステータス 200 | None |

### レスポンススキーマ

**注釈**

この操作には認証が必要ありません

## get_*server_group*{group_id}

### コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/server_group/{group_id} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}
```

(次のページに続く)

(前のページからの続き)

```
r = requests.get(https://PEM-SERVER-IP/pem/api/v17/server_group/{group_id}, headers =␣
↪headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch("https://PEM-SERVER-IP/pem/api/v17/server_group/{group_id}", {
  method: "GET",

  headers: headers,
})
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /server_group/{group_id}

*ID* でサーバーグループを取得します。

サーバーグループを返します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| group_id | path | integer | true | なし |
| X-Auth-Token | header | string | true | 認証用のトークン |

### 応答例

```
{
  "id": 0,
  "name": "string",
  "hidden": true,
  "parent_id": 0
}
```

応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | *ServerGroupBodyGet* |

**注釈**

この操作には認証が必要ありません

### put_*server_group*{group_id}

コードサンプル

```
#  You can also use wget

curl -X PUT https://PEM-SERVER-IP/pem/api/v17/server_group/{group_id} \
  -H Content-Type: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Content-Type: application/json,
  X-Auth-Token: string
}

r = requests.put(https://PEM-SERVER-IP/pem/api/v17/server_group/{group_id}, headers =␣
↪headers)

print(r.json())
```

```
const inputBody = {
  "name": "string"
};
const headers = {
  Content-Type:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v17/server_group/{group_id},
{
  method: PUT,
```

(次のページに続く)

(前のページからの続き)

```
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

PUT /server_group/{group_id}

サーバーグループを更新します。

ボディパラメーター

```
{
  "name": "string"
}
```

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| group_id | path | integer | true | なし |
| X-Auth-Token | header | string | true | 承認トークン |
| body | body | *Server-GroupBody* | true | 正常に更新されました |

### 応答例

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ステータス 200 | None |

### レスポンススキーマ

**注釈**

この操作には認証が必要ありません

### delete_*server_group*{group_id}

コードサンプル

```
#  You can also use wget

curl -X DELETE https://PEM-SERVER-IP/pem/api/v17/server_group/{group_id} \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  X-Auth-Token: string
}

r = requests.delete(https://PEM-SERVER-IP/pem/api/v17/server_group/{group_id},␣
↪headers = headers)

print(r.json())
```

```
const headers = {
  "X-Auth-Token": "string",
};

fetch("https://PEM-SERVER-IP/pem/api/v17/server_group/{group_id}", {
  method: "DELETE",

  headers: headers,
})
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

DELETE /server_group/{group_id}

サーバーグループを削除します。

#### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| group_id | path | integer | true | なし |
| X-Auth-Token | header | string | true | 認証用のトークン |

応答例

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ステータス 200 | None |

### レスポンススキーマ

> **注釈**
>
> この操作には認証が必要ありません

### get_*agent*

### コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/agent/ \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/agent/, headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch("https://PEM-SERVER-IP/pem/api/v17/agent/", {
  method: "GET",

  headers: headers,
})
```

(次のページに続く)

(前のページからの続き)

```
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /agent/

*pem-agent* 情報を取得します。

ユーザーに表示される pem エージェントの情報とステータスのリストを返しますトークンで表されます。name 完全一致または name_contains 部分文字列検索クエリパラメーターを介してオプションの名前ベースのフィルタリングをサポートします。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| X-Auth-Token | header | string | true | 認証用のトークン |
| name | query | string | false | 完全な名前の一致大文字と小文字を区別してエージェントをフィルタリングします。 name_contains と相互排他的。 |
| name_contains | query | string | false | 部分文字列の一致大文字と小文字を区別してエージェントをフィルタリングします。 name と相互排他的。 |

### 応答例

```
[
  {
    "id": 0,
    "name": "string",
    "version": "string",
    "profile_id": 0,
    "active": true,
    "heartbeat_tolerance": 0,
    "alert_blackout": true,
    "owner": "string",
    "team": "string",
    "gid": 0,
    "tags": ["string"],
    "ignore_mnt_points": ["string"]
```

(次のページに続く)

(前のページからの続き)

```
  }
]
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | pem エージェントに関する情報とステータスのリストを返します。 | *PEM* エージェントの証明書とキーファイル |
| 400 | Bad Request | 相互排他性違反 `name`または類似の完全一致フィルタと`name_contains`または類似の部分文字列フィルタが両方とも同じ要求で指定されました。フィルタリングは、ペアのうちの 1 つのみを受け入れます。 | Inline |

### レスポンススキーマ

ステータスコード **400**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » success | integer | false | none | なし |
| » errormsg | string | false | none | なし |

**注釈**

この操作には認証が必要ありません

### get_*agent*{agent_id}

### コードサンプル

```
#  You can also use wget

```

(次のページに続く)

(前のページからの続き)

```
curl -X GET https://PEM-SERVER-IP/pem/api/v17/agent/{agent_id} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/agent/{agent_id}, headers =␣
↪headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch("https://PEM-SERVER-IP/pem/api/v17/agent/{agent_id}", {
  method: "GET",

  headers: headers,
})
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /agent/{agent_id}

*agent_id* で識別されるエージェントの情報を取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| agent_id | path | integer | true | なし |
| X-Auth-Token | header | string | true | 承認トークン |

**応答例**

```
{
  "id": 0,
  "name": "string",
  "version": "string",
  "profile_id": 0,
  "active": true,
  "heartbeat_tolerance": 0,
  "alert_blackout": true,
  "owner": "string",
  "team": "string",
  "gid": 0,
  "tags": ["string"],
  "ignore_mnt_points": ["string"]
}
```

応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | *PEM* バックエンドデータベースサーバーとエージェントの接続証明書 |

**注釈**

この操作には認証が必要ありません

### put_*agent*{agent_id}

コードサンプル

```
#  You can also use wget

curl -X PUT https://PEM-SERVER-IP/pem/api/v17/agent/{agent_id} \
  -H Content-Type: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Content-Type: application/json,
  X-Auth-Token: string
}
```

(次のページに続く)

(前のページからの続き)

```
r = requests.put(https://PEM-SERVER-IP/pem/api/v17/agent/{agent_id}, headers =␣
↪headers)

print(r.json())
```

```
const inputBody = {
  "name": "string",
  "heartbeat_tolerance": 0,
  "alert_blackout": true,
  "profile_id": 0,
  "team": "string",
  "gid": 0,
  "tags": {
    "added": [
      {
        "color": "#0C767A",
        "text": "tag4"
      }
    ],
    "deleted": [
      {
        "text": "tag1",
        "color": "#8789AC",
        "old_text": "tag2"
      }
    ],
    "changed": [
      {
        "old_text": "tag",
        "text": "tag3"
      }
    ]
  },
  "ignore_mnt_points": [
    "string"
  ]
};
const headers = {
  Content-Type:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v17/agent/{agent_id},
{
```

(次のページに続く)

(前のページからの続き)

```
  method: PUT,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

PUT /agent/{agent_id}

エージェント情報の更新。

ボディパラメーター

```
{
  "name": "string",
  "heartbeat_tolerance": 0,
  "alert_blackout": true,
  "profile_id": 0,
  "team": "string",
  "gid": 0,
  "tags": {
    "added": [
      {
        "color": "#0C767A",
        "text": "tag4"
      }
    ],
    "deleted": [
      {
        "text": "tag1",
        "color": "#8789AC",
        "old_text": "tag2"
      }
    ],
    "changed": [
      {
        "old_text": "tag",
        "text": "tag3"
      }
    ]
  },
  "ignore_mnt_points": ["string"]
}
```

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| agent_id | path | integer | true | なし |
| X-Auth-Token | header | string | true | 承認トークン |
| body | body | *AgentBody* | true | なし |

#### 応答例

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 正常に更新されました | None |

### レスポンススキーマ

**注釈**

この操作には認証が必要ありません

## delete_*agent*{agent_id}

#### コードサンプル

```
#  You can also use wget

curl -X DELETE https://PEM-SERVER-IP/pem/api/v17/agent/{agent_id} \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  X-Auth-Token: string
}

r = requests.delete(https://PEM-SERVER-IP/pem/api/v17/agent/{agent_id}, headers =␣
↪headers)

print(r.json())
```

```
const headers = {
  "X-Auth-Token": "string",
};

fetch("https://PEM-SERVER-IP/pem/api/v17/agent/{agent_id}", {
  method: "DELETE",

  headers: headers,
})
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

`DELETE /agent/{agent_id}`

*PEM* によるエージェントの管理を停止します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| agent_id | path | integer | true | なし |
| X-Auth-Token | header | string | true | 承認トークン |

**応答例**

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ステータス 200 | None |

### レスポンススキーマ

**注釈**

この操作には認証が必要ありません

### get_*server*

コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/server/ \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/server/, headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch("https://PEM-SERVER-IP/pem/api/v17/server/", {
  method: "GET",

  headers: headers,
})
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /server/

#### 管理対象サーバーリストの取得

管理対象サーバーのリストを返します。 name 完全一致または name_contains 部分文字列検索クエリパラメーターを介してオプションの名前ベースのフィルタリングをサポートします。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| X-Auth-Token | header | string | true | 承認トークン |
| name | query | string | false | 完全な名前一致大文字と小文字を区別してサーバーをフィルタリングします。 name_contains と相互排他的。 |
| name_contains | query | string | false | 部分文字列の一致大文字と小文字を区別してサーバーをフィルタリングします。 name と相互排他的。 |

### 応答例


```
[
  {
    "id": 0,
    "name": "string",
    "host": "string",
    "port": 0,
    "db": "string",
    "profile_id": 0,
    "serviceid": "string",
    "connection_params": [
      {
        "name": "string",
        "keyword": "string",
        "value": null
      }
    ],
    "active": true,
    "alert_blackout": true,
    "owner": 0,
    "team": "string",
    "server_owner": "string",
    "is_remote_monitoring": true,
    "tags": ["string"],
    "replication_solution": "none",
    "efm_cluster_name": "string",
    "efm_service_name": "string",
    "efm_installation_path": "string",
    "patroni_cluster_name": "string",
    "patroni_installation_path": "string",
    "patroni_config_path": "string",
    "comment": "string",
```


(次のページに続く)

(前のページからの続き)


```
    "username": "string",
    "gid": 0,
    "server_group_name": "string",
    "db_res": "string",
    "role": "string",
    "is_edb": true,
    "agent_id": 0,
    "asb_host": "string",
    "asb_port": 0,
    "asb_username": "string",
    "asb_database": "string",
    "asb_sslmode": "prefer",
    "agent_allowtakeover": false,
    "agent_capability_list": ["string"],
    "agent_description": "string",
    "sslcompression": false,
    "use_ssh_tunnel": false,
    "tunnel_host": "string",
    "tunnel_port": 22,
    "tunnel_username": "string",
    "tunnel_authentication": false,
    "tunnel_identity_file": "string",
    "tunnel_keep_alive": 0,
    "post_connection_sql": "string"
  }
]
```


### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ステータス 200 | Inline |

次のページに続く

表 82 – 前のページからの続き

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 400 | Bad Request | 相互排他性違反 `name`または類似の完全一致フィルタと`name_contains`または類似の部分文字列フィルタが両方とも同じ要求で指定されました。フィルタリングは、ペアのうちの 1 つのみを受け入れます。 | Inline |

### レスポンススキーマ

ステータスコード **200**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| _anonymous_ | *Platform Compatibility* ] | false | none | なし |
| » id | integer | false | none | サーバー ID. 注 このプロパティは、POST 要求には適用されません。 |
| » name | string | false | none | サーバー名 |
| » host | string | false | none | サーバーホスト |
| » port | integer | false | none | サーバーポート |
| » db | string | false | none | メンテナンスデータベース |
| » profile_id | integer | false | none | プロファイル ID |
| » serviceid | string | false | none | サーバーサービス ID |
| » connection_params | [object] | false | none | サーバーの接続パラメーターのリスト。 |
| » name | string | false | none | パラメーター名 |
| » keyword | string | false | none | パラメーターキーワード |
| » value | any | false | none | パラメーター値 |
| » active | boolean | false | none | サーバーがアクティブかどうかを判断します。 |
| » alert_blackout | boolean | false | none | サーバー上のアラートがブラックアウトされているかどうかを判断します。 |

次のページに続く

表 83 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » owner | integer | false | none | サーバーを PEM に登録した所有者の ID。 |
| » team | string | false | none | サーバー情報を表示できるロール/ユーザー名を定義します。 |
| » server_owner | string | false | none | PEM データベースサーバーにサーバーを登録したユーザーの名前。 |
| » is_remote_monitoring | boolean | false | none | このサーバーがリモートで監視されているかどうかを判断します。つまり、監視 pem エージェントが監視対象サーバーと同じホストにありません。 |
| » tags | [string] | false | none | サーバーに関連付けられたタグのリスト。 |
| » replication_solution | string | false | none | サーバーが使用するレプリケーションソリューション |
| » efm_cluster_name | string | false | none | EFM のクラスター名 |
| » efm_service_name | string | false | none | EFM のサービス名 |
| » efm_installation_path | string | false | none | EFM のインストールパス |
| » patroni_cluster_name | string | false | none | Patroni のクラスター名 |
| » patroni_installation_path | string | false | none | Patroni のインストールパス |
| » patroni_config_path | string | false | none | Patroni の構成パス |
| » comment | string | false | none | サーバーの説明 |
| » username | string | false | none | データベースサーバーに接続するためのユーザー名 |
| » gid | integer | false | none | サーバーグループの ID |
| » server_group_name | string | false | none | サーバーグループの名前 |
| » db_res | string | false | none | データベース制限 SQL 句 |
| » role | string | false | none | 接続が使用するロール |
| » is_edb | boolean | false | none | サーバーは EDB Advanced Server ですか？ |
| » agent_id | integer | false | none | サーバーにバインドされたエージェント ID |

次のページに続く

表 83 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » asb_host | string | false | none | PEM 接続の作成中にエージェントが使用する PEM データベースホスト |
| » asb_port | integer | false | none | PEM 接続の作成中にエージェントが使用する PEM データベースポート |
| » asb_username | string | false | none | PEM 接続の作成中にエージェントが使用する PEM データベースのユーザー名 |
| » asb_database | string | false | none | エージェントが接続する PEM データベース |
| » asb_sslmode | string | false | none | PEM データベース SSL モード |
| » agent_allowtakeover | boolean | false | none | サーバーを別のエージェントに引き継ぎできるかどうかを指定します。 |
| » agent_capability_list | [string] | false | none | エージェント機能の配列 |
| » agent_description | string | false | none | エージェントの説明 |
| » sslcompression | boolean | false | none | SSL 経由で送信されるデータを圧縮するかどうかを決定します。 |
| » use_ssh_tunnel | boolean | false | none | ssh トンネルを使用するかどうかを決定します。 |
| » tunnel_host | string | false | none | トンネルホスト名 |
| » tunnel_port | integer | false | none | トンネルポート番号 |
| » tunnel_username | string | false | none | トンネルのユーザー名 |
| » tunnel_authentication | boolean | false | none | トンネル認証を使用するかどうかを決定します。 |
| » tunnel_identity_file | string | false | none | 認証に使用されるトンネル ID ファイル。 |
| » tunnel_keep_alive | integer | false | none | ssh トンネルのキープアライブパケットを送信する秒単位の間隔。 |
| » post_connection_sql | string | false | none | 接続の確立後に実行される SQL 。 |

#### 列挙値

| Property | Value |
|---|---|
| replication_solution | none |
| replication_solution | efm |
| replication_solution | patroni |
| asb_sslmode | prefer |
| asb_sslmode | allow |
| asb_sslmode | disable |
| asb_sslmode | require |
| asb_sslmode | verify-ca |
| asb_sslmode | verify-full |

ステータスコード **400**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » success | integer | false | none | なし |
| » errormsg | string | false | none | なし |

**注釈**

この操作には認証が必要ありません

### post_*server*

#### コードサンプル

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v17/server/ \
  -H Content-Type: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Content-Type: application/json,
  X-Auth-Token: string
}

r = requests.post(https://PEM-SERVER-IP/pem/api/v17/server/, headers = headers)

print(r.json())
```

```
const inputBody = {
  "name": "string",
  "host": "string",
  "port": 0,
  "db": "string",
  "profile_id": 0,
  "serviceid": "string",
  "connection_params": [
    {
      "name": "sslmode",
      "keyword": "sslmode",
      "value": "prefer"
    },
    {
      "name": "connect_timeout",
      "keyword": "connect_timeout",
      "value": 10
    }
  ],
  "alert_blackout": true,
  "team": "string",
  "is_remote_monitoring": true,
  "tags": [
    {
      "text": "tag1",
      "color": "#8789AC"
    },
    {
      "text": "tag2",
      "color": "#0C767A"
    }
  ],
  "replication_solution": "none",
  "efm_cluster_name": "string",
  "efm_service_name": "string",
  "efm_installation_path": "string",
  "patroni_cluster_name": "string",
  "patroni_installation_path": "string",
  "patroni_config_path": "string",
  "comment": "string",
  "username": "string",
  "gid": 0,
  "hostaddr": "string",
  "db_res": "string",
  "role": "string",
```

(次のページに続く)

(前のページからの続き)

```
  "agent_id": 0,
  "asb_host": "string",
  "asb_port": 0,
  "asb_username": "string",
  "asb_database": "string",
  "asb_sslmode": "prefer",
  "asb_password": "string",
  "agent_allowtakeover": false,
  "fgcolor": "#000000",
  "bgcolor": "#FFFFFF",
  "sslcompression": false,
  "use_ssh_tunnel": false,
  "tunnel_host": "string",
  "tunnel_port": 22,
  "tunnel_username": "string",
  "tunnel_password": "string",
  "tunnel_authentication": false,
  "tunnel_identity_file": "string",
  "tunnel_keep_alive": 0,
  "post_connection_sql": "string"
};
const headers = {
  Content-Type:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v17/server/,
{
  method: POST,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

POST /server/

新しい管理対象データベースサーバーを追加します。

ボディパラメーター

```
{
  "name": "string",
```

(次のページに続く)

(前のページからの続き)


```
  "host": "string",
  "port": 0,
  "db": "string",
  "profile_id": 0,
  "serviceid": "string",
  "connection_params": [
    {
      "name": "sslmode",
      "keyword": "sslmode",
      "value": "prefer"
    },
    {
      "name": "connect_timeout",
      "keyword": "connect_timeout",
      "value": 10
    }
  ],
  "alert_blackout": true,
  "team": "string",
  "is_remote_monitoring": true,
  "tags": [
    {
      "text": "tag1",
      "color": "#8789AC"
    },
    {
      "text": "tag2",
      "color": "#0C767A"
    }
  ],
  "replication_solution": "none",
  "efm_cluster_name": "string",
  "efm_service_name": "string",
  "efm_installation_path": "string",
  "patroni_cluster_name": "string",
  "patroni_installation_path": "string",
  "patroni_config_path": "string",
  "comment": "string",
  "username": "string",
  "gid": 0,
  "hostaddr": "string",
  "db_res": "string",
  "role": "string",
  "agent_id": 0,
  "asb_host": "string",
```


(次のページに続く)

(前のページからの続き)


```
  "asb_port": 0,
  "asb_username": "string",
  "asb_database": "string",
  "asb_sslmode": "prefer",
  "asb_password": "string",
  "agent_allowtakeover": false,
  "fgcolor": "#000000",
  "bgcolor": "#FFFFFF",
  "sslcompression": false,
  "use_ssh_tunnel": false,
  "tunnel_host": "string",
  "tunnel_port": 22,
  "tunnel_username": "string",
  "tunnel_password": "string",
  "tunnel_authentication": false,
  "tunnel_identity_file": "string",
  "tunnel_keep_alive": 0,
  "post_connection_sql": "string"
}
```


### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| X-Auth-Token | header | string | true | 承認トークン |
| body | body | *ServerBody* | true | なし |

### 応答例

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ステータス 200 | None |

### レスポンススキーマ

**注釈**

この操作には認証が必要ありません

### get_*server*{server_id}

コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/server/{server_id} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/server/{server_id}, headers =␣
↪headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch("https://PEM-SERVER-IP/pem/api/v17/server/{server_id}", {
  method: "GET",

  headers: headers,
})
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /server/{server_id}

*server_id* で識別されるサーバーの情報を取得します。

#### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | string | true | なし |
| X-Auth-Token | header | string | true | 承認トークン |

**応答例**


```
{
  "id": 0,
  "name": "string",
  "host": "string",
  "port": 0,
  "db": "string",
  "profile_id": 0,
  "serviceid": "string",
  "connection_params": [
    {
      "name": "string",
      "keyword": "string",
      "value": null
    }
  ],
  "active": true,
  "alert_blackout": true,
  "owner": 0,
  "team": "string",
  "server_owner": "string",
  "is_remote_monitoring": true,
  "tags": ["string"],
  "replication_solution": "none",
  "efm_cluster_name": "string",
  "efm_service_name": "string",
  "efm_installation_path": "string",
  "patroni_cluster_name": "string",
  "patroni_installation_path": "string",
  "patroni_config_path": "string",
  "comment": "string",
  "username": "string",
  "gid": 0,
  "server_group_name": "string",
  "db_res": "string",
  "role": "string",
  "is_edb": true,
  "agent_id": 0,
  "asb_host": "string",
  "asb_port": 0,
```


(次のページに続く)

(前のページからの続き)

```
  "asb_username": "string",
  "asb_database": "string",
  "asb_sslmode": "prefer",
  "agent_allowtakeover": false,
  "agent_capability_list": ["string"],
  "agent_description": "string",
  "sslcompression": false,
  "use_ssh_tunnel": false,
  "tunnel_host": "string",
  "tunnel_port": 22,
  "tunnel_username": "string",
  "tunnel_authentication": false,
  "tunnel_identity_file": "string",
  "tunnel_keep_alive": 0,
  "post_connection_sql": "string"
}
```

応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 正常に作成されました | *Web* サーバー証明書 |

**注釈**

この操作には認証が必要ありません

### put_*server*{server_id}

コードサンプル

```
#  You can also use wget

curl -X PUT https://PEM-SERVER-IP/pem/api/v17/server/{server_id} \
  -H Content-Type: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Content-Type: application/json,
```

(次のページに続く)

(前のページからの続き)

```
  X-Auth-Token: string
}

r = requests.put(https://PEM-SERVER-IP/pem/api/v17/server/{server_id}, headers =␣
↪headers)

print(r.json())
```

```
const inputBody = {
  "name": "string",
  "host": "string",
  "port": 0,
  "db": "string",
  "profile_id": 0,
  "serviceid": "string",
  "connection_params": {
    "added": [
      {
        "keyword": "hostaddr",
        "name": "hostaddr",
        "value": "qwde"
      }
    ],
    "changed": [
      {
        "keyword": "connect_timeout",
        "name": "connect_timeout",
        "value": "20"
      }
    ],
    "deleted": [
      {
        "keyword": "sslmode",
        "name": "sslmode",
        "value": "prefer"
      }
    ]
  },
  "alert_blackout": true,
  "team": "string",
  "is_remote_monitoring": true,
  "tags": {
    "added": [
      {
```

(次のページに続く)

(前のページからの続き)

```
        "color": "#0C767A",
        "text": "tag4"
      }
    ],
    "deleted": [
      {
        "text": "tag1",
        "color": "#8789AC",
        "old_text": "tag2"
      }
    ],
    "changed": [
      {
        "old_text": "tag",
        "text": "tag3"
      }
    ]
  },
  "replication_solution": "none",
  "efm_cluster_name": "string",
  "efm_service_name": "string",
  "efm_installation_path": "string",
  "patroni_cluster_name": "string",
  "patroni_installation_path": "string",
  "patroni_config_path": "string",
  "comment": "string",
  "username": "string",
  "gid": 0,
  "hostaddr": "string",
  "db_res": "string",
  "role": "string",
  "agent_id": 0,
  "asb_host": "string",
  "asb_port": 0,
  "asb_username": "string",
  "asb_database": "string",
  "asb_sslmode": "prefer",
  "asb_password": "string",
  "agent_allowtakeover": false,
  "fgcolor": "#000000",
  "bgcolor": "#FFFFFF",
  "sslcompression": false,
  "use_ssh_tunnel": false,
  "tunnel_host": "string",
  "tunnel_port": 22,
```

(次のページに続く)

(前のページからの続き)

```
  "tunnel_username": "string",
  "tunnel_password": "string",
  "tunnel_authentication": false,
  "tunnel_identity_file": "string",
  "tunnel_keep_alive": 0,
  "post_connection_sql": "string"
};
const headers = {
  Content-Type:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v17/server/{server_id},
{
  method: PUT,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

PUT /server/{server_id}

サーバー情報を更新します。

ボディパラメーター

```
{
  "name": "string",
  "host": "string",
  "port": 0,
  "db": "string",
  "profile_id": 0,
  "serviceid": "string",
  "connection_params": {
    "added": [
      {
        "keyword": "hostaddr",
        "name": "hostaddr",
        "value": "qwde"
      }
    ],
    "changed": [
```

(次のページに続く)

(前のページからの続き)

```
      {
        "keyword": "connect_timeout",
        "name": "connect_timeout",
        "value": "20"
      }
    ],
    "deleted": [
      {
        "keyword": "sslmode",
        "name": "sslmode",
        "value": "prefer"
      }
    ]
  },
  "alert_blackout": true,
  "team": "string",
  "is_remote_monitoring": true,
  "tags": {
    "added": [
      {
        "color": "#0C767A",
        "text": "tag4"
      }
    ],
    "deleted": [
      {
        "text": "tag1",
        "color": "#8789AC",
        "old_text": "tag2"
      }
    ],
    "changed": [
      {
        "old_text": "tag",
        "text": "tag3"
      }
    ]
  },
  "replication_solution": "none",
  "efm_cluster_name": "string",
  "efm_service_name": "string",
  "efm_installation_path": "string",
  "patroni_cluster_name": "string",
  "patroni_installation_path": "string",
  "patroni_config_path": "string",
```

(次のページに続く)

(前のページからの続き)


```
  "comment": "string",
  "username": "string",
  "gid": 0,
  "hostaddr": "string",
  "db_res": "string",
  "role": "string",
  "agent_id": 0,
  "asb_host": "string",
  "asb_port": 0,
  "asb_username": "string",
  "asb_database": "string",
  "asb_sslmode": "prefer",
  "asb_password": "string",
  "agent_allowtakeover": false,
  "fgcolor": "#000000",
  "bgcolor": "#FFFFFF",
  "sslcompression": false,
  "use_ssh_tunnel": false,
  "tunnel_host": "string",
  "tunnel_port": 22,
  "tunnel_username": "string",
  "tunnel_password": "string",
  "tunnel_authentication": false,
  "tunnel_identity_file": "string",
  "tunnel_keep_alive": 0,
  "post_connection_sql": "string"
}
```


### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | string | true | なし |
| X-Auth-Token | header | string | true | 承認トークン |
| body | body | *Server-BodyUpdate* | true | なし |

### 応答例

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 正常に更新されました。 | None |

レスポンススキーマ

> **注釈**
>
> この操作には認証が必要ありません

### delete_*server*{server_id}

コードサンプル

```
#  You can also use wget

curl -X DELETE https://PEM-SERVER-IP/pem/api/v17/server/{server_id} \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  X-Auth-Token: string
}

r = requests.delete(https://PEM-SERVER-IP/pem/api/v17/server/{server_id}, headers =␣
↪headers)

print(r.json())
```

```
const headers = {
  "X-Auth-Token": "string",
};

fetch("https://PEM-SERVER-IP/pem/api/v17/server/{server_id}", {
  method: "DELETE",

  headers: headers,
})
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
```

(次のページに続く)

(前のページからの続き)

```
  console.log(body);
});
```

```
DELETE /server/{server_id}
```

*PEM* によるサーバーの管理を停止します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | string | true | なし |
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ステータス 200 | None |

### レスポンススキーマ

**注釈**

この操作には認証が必要ありません

## get_*server*{sid}*database*

### コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/server/{sid}/database/ \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
```

(次のページに続く)

(前のページからの続き)

```
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/server/{sid}/database/, headers =␣
↪headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch("https://PEM-SERVER-IP/pem/api/v17/server/{sid}/database/", {
  method: "GET",

  headers: headers,
})
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /server/{sid}/database/

管理対象サーバーのデータベースのプロパティのリストを取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| X-Auth-Token | header | string | true | 承認トークン |
| sid | path | integer | true | サーバー ID |

### 応答例

```
[
  {
    "database_name": "string",
    "connections_allowed": true,
    "system_database": true,
    "recorded_time": "2019-08-24T14:15:22Z",
    "encoding": "string",
```

(次のページに続く)

(前のページからの続き)

```
    "server_id": 0
  }
]
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ステータス 200 | Inline |

### レスポンススキーマ

ステータスコード **200**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| _anonymous_ | *Platform Compatibility* ] | false | none | なし |
| » database_name | string | false | none | データベース名 |
| » connections_allowed | boolean | false | none | データベースへの接続を許可するかどうかを決定します。 |
| » system_database | boolean | false | none | データベースがシステムデータベースであるかどうかを判断します。 |
| » recorded_time | string(date-time) | false | none | データベースが PEM エージェントによって最初に検出されたときを指定します。 |
| » encoding | string | false | none | データベースエンコーディング |
| » server_id | integer | false | none | データベースのサーバー ID |

**注釈**

この操作には認証が必要ありません

### get_*server*{sid}*database*{db_name}

コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/server/{sid}/database/{db_name} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/server/{sid}/database/{db_name},␣
↪headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch("https://PEM-SERVER-IP/pem/api/v17/server/{sid}/database/{db_name}", {
  method: "GET",

  headers: headers,
})
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /server/{sid}/database/{db_name}

管理対象サーバーの特定のデータベースのプロパティを取得します。

パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| sid | path | integer | true | サーバー ID |
| db_name | path | string | true | なし |
| X-Auth-Token | header | string | true | 承認トークン |

#### 応答例

```
{
  "database_name": "string",
  "connections_allowed": true,
  "system_database": true,
  "recorded_time": "2019-08-24T14:15:22Z",
  "encoding": "string",
  "server_id": 0
}
```

#### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ステータス 200 | *PEM* バックエンドデータベースサーバーとエージェントの接続証明書 |

**注釈**

この操作には認証が必要ありません

### get_*server*{sid}*database*{db_name}*schema*

#### コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/server/{sid}/database/{db_name}/schema/␣
↪\
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
```

(次のページに続く)

(前のページからの続き)

```
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/server/{sid}/database/{db_name}/
↪schema/, headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch(
  "https://PEM-SERVER-IP/pem/api/v17/server/{sid}/database/{db_name}/schema/",
  {
    method: "GET",

    headers: headers,
  },
)
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /server/{sid}/database/{db_name}/schema/

管理対象サーバーの特定のデータベースのスキーマのプロパティのリストを取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| sid | path | integer | true | サーバー ID |
| db_name | path | string | true | データベース名 |
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例

```
[
  {
    "schema_name": "string",
    "database_name": "string",
```

(次のページに続く)

(前のページからの続き)

```
    "server_id": 0,
    "recorded_time": "2019-08-24T14:15:22Z"
  }
]
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ステータス 200 | Inline |

### レスポンススキーマ

ステータスコード **200**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| _anonymous_ | *Platform Compatibility* ] | false | none | なし |
| » schema_name | string | false | none | スキーマ名 |
| » database_name | string | false | none | スキーマのデータベース名 |
| » server_id | integer | false | none | サーバー ID スキーマ |
| » recorded_time | string(date-time) | false | none | スキーマが PEM エージェントによって最初に検出されたときを指定します。 |

**注釈**

この操作には認証が必要ありません

### get_*server*{sid}*database*{db_name}*schema*{schema_name}

### コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/server/{sid}/database/{db_name}/schema/
↪{schema_name} \
```

(次のページに続く)

(前のページからの続き)

```
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/server/{sid}/database/{db_name}/
↪schema/{schema_name}, headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch(
  "https://PEM-SERVER-IP/pem/api/v17/server/{sid}/database/{db_name}/schema/{schema_
↪name}",
  {
    method: "GET",

    headers: headers,
  },
)
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

`GET /server/{sid}/database/{db_name}/schema/{schema_name}`

**管理対象サーバーの特定のデータベースのスキーマのプロパティを取得します。**

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| sid | path | integer | true | サーバー ID |
| db_name | path | string | true | データベース名 |

次のページに続く

表 102 – 前のページからの続き

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| schema_name | path | string | true | スキーマ名 |
| X-Auth-Token | header | string | true | 承認トークン |

#### 応答例

```
{
  "schema_name": "string",
  "database_name": "string",
  "server_id": 0,
  "recorded_time": "2019-08-24T14:15:22Z"
}
```

#### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ステータス 200 | *Using the Schema Diff tool* |

**注釈**

この操作には認証が必要ありません

### get_*server*{sid}*database*{db_name}*schema*{schema_name}*table*

#### コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/server/{sid}/database/{db_name}/schema/
↪{schema_name}/table/ \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}
```

(次のページに続く)

(前のページからの続き)

```
r = requests.get(https://PEM-SERVER-IP/pem/api/v17/server/{sid}/database/{db_name}/
↪schema/{schema_name}/table/, headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch(
  "https://PEM-SERVER-IP/pem/api/v17/server/{sid}/database/{db_name}/schema/{schema_
↪name}/table/",
  {
    method: "GET",

    headers: headers,
  },
)
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /server/{sid}/database/{db_name}/schema/{schema_name}/table/

**管理対象サーバーの特定のデータベースのスキーマにあるテーブルのプロパティのリストを取得します。**

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| sid | path | integer | true | サーバー ID |
| db_name | path | string | true | データベース名 |
| schema_name | path | string | true | スキーマ名 |
| X-Auth-Token | header | string | true | 承認トークン |

**応答例**

```
[
  {
    "table_name": "string",
    "schema_name": "string",
```

(次のページに続く)

(前のページからの続き)

```
    "database_name": "string",
    "server_id": 0,
    "has_primary_key": true,
    "recorded_time": "string"
  }
]
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ステータス 200 | Inline |

### レスポンススキーマ

ステータスコード **200**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| _anonymous_ | *Platform Compatibility* ] | false | none | なし |
| » table_name | string | false | none | テーブル名 |
| » schema_name | string | false | none | スキーマ名 |
| » database_name | string | false | none | データベース名 |
| » server_id | integer | false | none | サーバー ID |
| » has_primary_key | boolean | false | none | テーブルに主キーがあるかどうかを判断します。 |
| » recorded_time | string | false | none | テーブルが PEM エージェントによって最初に検出されたときを指定します。 |

**注釈**

この操作には認証が必要ありません

### get_*server*{sid}*database*{db_name}*schema*{schema_name}*table*{table_name}

コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/server/{sid}/database/{db_name}/schema/
↪{schema_name}/table/{table_name} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/server/{sid}/database/{db_name}/
↪schema/{schema_name}/table/{table_name}, headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch(
  "https://PEM-SERVER-IP/pem/api/v17/server/{sid}/database/{db_name}/schema/{schema_
↪name}/table/{table_name}",
  {
    method: "GET",

    headers: headers,
  },
)
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

```
GET /server/{sid}/database/{db_name}/schema/{schema_name}/table/{table_name}
```

管理対象サーバーの特定のデータベースのスキーマにあるテーブルのプロパティを取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| sid | path | integer | true | サーバー ID |
| db_name | path | string | true | データベース名 |
| schema_name | path | string | true | スキーマ名 |
| table_name | path | string | true | なし |
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例

```
{
  "table_name": "string",
  "schema_name": "string",
  "database_name": "string",
  "server_id": 0,
  "has_primary_key": true,
  "recorded_time": "string"
}
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ステータス 200 | テーブルリレーションシップオプション |

**注釈**

この操作には認証が必要ありません

### get_*server*{sid}*database*{db_name}*schema*{schema_name}*index*

### コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/server/{sid}/database/{db_name}/schema/
↪{schema_name}/index/ \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/server/{sid}/database/{db_name}/
↪schema/{schema_name}/index/, headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch(
  "https://PEM-SERVER-IP/pem/api/v17/server/{sid}/database/{db_name}/schema/{schema_
↪name}/index/",
  {
    method: "GET",

    headers: headers,
  },
)
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /server/{sid}/database/{db_name}/schema/{schema_name}/index/

管理対象サーバーの特定のデータベースのスキーマにあるインデックスのプロパティのリストを取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| sid | path | integer | true | サーバー ID |
| db_name | path | string | true | データベース名 |
| schema_name | path | string | true | スキーマ名 |
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例

```
[
  {
    "index_name": "string",
    "ind_keys": [0],
    "table_name": "string",
    "schema_name": "string",
    "database_name": "string",
    "server_id": 0,
    "recorded_time": "string"
  }
]
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ステータス 200 | Inline |

### レスポンススキーマ

ステータスコード **200**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| _anonymous_ | *Platform Compatibility* ] | false | none | なし |
| » index_name | string | false | none | インデックス名 |
| » ind_keys | [integer] | false | none | テーブル内のインデックス付き列の位置 |
| » table_name | string | false | none | テーブル名 |
| » schema_name | string | false | none | スキーマ |
| » database_name | string | false | none | データベース名 |
| » server_id | integer | false | none | サーバー ID |
| » recorded_time | string | false | none | インデックスが PEM エージェントによって最初に検出されたときを指定します。 |

> **注釈**
>
> この操作には認証が必要ありません

### get_*server*{sid}*database*{db_name}*schema*{schema_name}*index*{index_name}

コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/server/{sid}/database/{db_name}/schema/
↪{schema_name}/index/{index_name} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/server/{sid}/database/{db_name}/
↪schema/{schema_name}/index/{index_name}, headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch(
  "https://PEM-SERVER-IP/pem/api/v17/server/{sid}/database/{db_name}/schema/{schema_
↪name}/index/{index_name}",
  {
    method: "GET",

    headers: headers,
  },
)
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

`GET /server/{sid}/database/{db_name}/schema/{schema_name}/index/{index_name}`

管理対象サーバーの特定のデータベースのスキーマにあるインデックスのプロパティを取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| sid | path | integer | true | サーバー ID |
| db_name | path | string | true | データベース名 |
| schema_name | path | string | true | スキーマ名 |
| index_name | path | string | true | なし |
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例

```
{
  "index_name": "string",
  "ind_keys": [0],
  "table_name": "string",
  "schema_name": "string",
  "database_name": "string",
  "server_id": 0,
  "recorded_time": "string"
}
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ステータス 200 | get__alert_config_server_{server_id}_database_{databas |

**注釈**

この操作には認証が必要ありません

## get_*server*{sid}*database*{db_name}*schema*{schema_name}*sequence*

### コードサンプル

```
#  You can also use wget
```

(次のページに続く)

(前のページからの続き)

```
curl -X GET https://PEM-SERVER-IP/pem/api/v17/server/{sid}/database/{db_name}/schema/
↪{schema_name}/sequence/ \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/server/{sid}/database/{db_name}/
↪schema/{schema_name}/sequence/, headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch(
  "https://PEM-SERVER-IP/pem/api/v17/server/{sid}/database/{db_name}/schema/{schema_
↪name}/sequence/",
  {
    method: "GET",

    headers: headers,
  },
)
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /server/{sid}/database/{db_name}/schema/{schema_name}/sequence/

管理対象サーバーの特定のデータベースのスキーマにあるシーケンスのプロパティのリストを取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| sid | path | integer | true | サーバー ID |
| db_name | path | string | true | データベース名 |
| schema_name | path | string | true | スキーマ名 |
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例

```
[
  {
    "sequence_name": "string",
    "schema_name": "string",
    "database_name": "string",
    "server_id": 0,
    "recorded_time": "2019-08-24T14:15:22Z"
  }
]
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ステータス 200 | Inline |

### レスポンススキーマ

ステータスコード **200**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| _anonymous_ | *Platform Compatibility* ] | false | none | なし |
| » sequence_name | string | false | none | シークエンス名 |
| » schema_name | string | false | none | スキーマ名 |
| » database_name | string | false | none | データベース名 |
| » server_id | integer | false | none | サーバー ID |

次のページに続く

表 116 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » recorded_time | string(date-time) | false | none | シーケンスが PEM エージェントによって最初に検出されたときを指定します。 |

**注釈**

この操作には認証が必要ありません

### get_*server*{sid}*database*{db_name}*schema*{schema_name}*sequence*{sequence_name}

#### コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/server/{sid}/database/{db_name}/schema/
↪{schema_name}/sequence/{sequence_name} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/server/{sid}/database/{db_name}/
↪schema/{schema_name}/sequence/{sequence_name}, headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch(
  "https://PEM-SERVER-IP/pem/api/v17/server/{sid}/database/{db_name}/schema/{schema_
↪name}/sequence/{sequence_name}",
  {
    method: "GET",
```

(次のページに続く)

(前のページからの続き)

```
    headers: headers,
  },
)
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

`GET /server/{sid}/database/{db_name}/schema/{schema_name}/sequence/{sequence_name}`

**管理対象サーバーの特定のデータベースのスキーマにあるシーケンスのプロパティを取得します。**

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| sid | path | integer | true | サーバー ID |
| db_name | path | string | true | データベース名 |
| schema_name | path | string | true | スキーマ名 |
| sequence_name | path | string | true | なし |
| X-Auth-Token | header | string | true | 承認トークン |

**応答例**

```
{
  "sequence_name": "string",
  "schema_name": "string",
  "database_name": "string",
  "server_id": 0,
  "recorded_time": "2019-08-24T14:15:22Z"
}
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ステータス 200 | get__alert_config_server_{server_id}_database_{databas |

> **注釈**
>
> この操作には認証が必要ありません

### get_*server*{sid}*database*{db_name}*schema*{schema_name}*view*

コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/server/{sid}/database/{db_name}/schema/
↪{schema_name}/view/ \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/server/{sid}/database/{db_name}/
↪schema/{schema_name}/view/, headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch(
  "https://PEM-SERVER-IP/pem/api/v17/server/{sid}/database/{db_name}/schema/{schema_
↪name}/view/",
  {
    method: "GET",

    headers: headers,
  },
)
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

`GET /server/{sid}/database/{db_name}/schema/{schema_name}/view/`

管理対象サーバーの特定のデータベースのスキーマにあるビューのプロパティのリストを取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| sid | path | integer | true | サーバー ID |
| db_name | path | string | true | データベース名 |
| schema_name | path | string | true | スキーマ名 |
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例


```
[
  {
    "view_name": "string",
    "view_type": "string",
    "ispopulated": true,
    "view_owner": "string",
    "definition": "string",
    "tablespace_name": "string",
    "schema_name": "string",
    "database_name": "string",
    "server_id": 0,
    "recorded_time": "string"
  }
]
```


### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ステータス 200 | Inline |

### レスポンススキーマ

ステータスコード **200**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| _anonymous_ | *Platform Compatibility* ] | false | none | なし |
| » view_name | string | false | none | ビュー名 |
| » view_type | string | false | none | ビュータイプ |
| » ispopulated | boolean | false | none | ビューが設定されているかどうかを判断します。 |
| » view_owner | string | false | none | ビューの所有者 |
| » definition | string | false | none | ビュー定義 |
| » tablespace_name | string | false | none | ビューテーブルスペース |
| » schema_name | string | false | none | ビュースキーマ |
| » database_name | string | false | none | データベースを見る |
| » server_id | integer | false | none | ビューサーバー ID |
| » recorded_time | string | false | none | ビューが PEM エージェントによって最初に検出されたときを指定します。 |

**注釈**

この操作には認証が必要ありません

### get_*server*{sid}*database*{db_name}*schema*{schema_name}*view*{view_name}

#### コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/server/{sid}/database/{db_name}/schema/
↪{schema_name}/view/{view_name} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/server/{sid}/database/{db_name}/
↪schema/{schema_name}/view/{view_name}, headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch(
  "https://PEM-SERVER-IP/pem/api/v17/server/{sid}/database/{db_name}/schema/{schema_
↪name}/view/{view_name}",
  {
    method: "GET",

    headers: headers,
  },
)
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /server/{sid}/database/{db_name}/schema/{schema_name}/view/{view_name}

管理対象サーバーの特定のデータベースのスキーマにあるビューのプロパティを取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| sid | path | integer | true | サーバー ID |
| db_name | path | string | true | データベース名 |
| schema_name | path | string | true | スキーマ名 |
| view_name | path | string | true | なし |
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例

```
{
  "view_name": "string",
  "view_type": "string",
  "ispopulated": true,
  "view_owner": "string",
  "definition": "string",
  "tablespace_name": "string",
  "schema_name": "string",
  "database_name": "string",
```

(次のページに続く)

(前のページからの続き)

```
  "server_id": 0,
  "recorded_time": "string"
}
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ステータス 200 | *Reviewing or modifying agent properties* |

**注釈**

この操作には認証が必要ありません

### get_*server*{sid}*database*{db_name}*schema*{schema_name}*function*

#### コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/server/{sid}/database/{db_name}/schema/
↪{schema_name}/function/ \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/server/{sid}/database/{db_name}/
↪schema/{schema_name}/function/, headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};
```

(次のページに続く)

(前のページからの続き)

```
fetch(
  "https://PEM-SERVER-IP/pem/api/v17/server/{sid}/database/{db_name}/schema/{schema_
↪name}/function/",
  {
    method: "GET",

    headers: headers,
  },
)
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /server/{sid}/database/{db_name}/schema/{schema_name}/function/

管理対象サーバーの特定のデータベースのスキーマにあるファンクションのプロパティのリストを取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| sid | path | integer | true | サーバー ID |
| db_name | path | string | true | データベース名 |
| schema_name | path | string | true | スキーマ名 |
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例

```
[
  {
    "function_name": "string",
    "function_type": "string",
    "return_type": "string",
    "arg_types": ["string"],
    "function_binary": "string",
    "extension_name": "string",
    "package_name": "string",
    "schema_name": "string",
    "database_name": "string",
    "server_id": 0,
```

(次のページに続く)

(前のページからの続き)

```
    "recorded_time": "2019-08-24T14:15:22Z"
  }
]
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ステータス 200 | Inline |

### レスポンススキーマ

ステータスコード **200**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| _anonymous_ | *Platform Compatibility* ] | false | none | なし |
| » function_name | string | false | none | ファンクション名 |
| » function_type | string | false | none | ファンクションタイプ |
| » return_type | string | false | none | ファンクションの戻り値の型 |
| » arg_types | [string] | false | none | ファンクションの引数の型 |
| » function_binary | string | false | none | なし |
| » extension_name | string | false | none | なし |
| » package_name | string | false | none | ファンクションパッケージ名 |
| » schema_name | string | false | none | ファンクションスキーマ |
| » database_name | string | false | none | ファンクションデータベース名 |
| » server_id | integer | false | none | ファンクションサーバー ID |
| » recorded_time | string(date-time) | false | none | ファンクションが PEM エージェントによって最初に検出されたときを指定します。 |

**注釈**

この操作には認証が必要ありません

### get_*server*{sid}*database*{db_name}*schema*{schema_name}*function*{function_name}

コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/server/{sid}/database/{db_name}/schema/
↪{schema_name}/function/{function_name} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/server/{sid}/database/{db_name}/
↪schema/{schema_name}/function/{function_name}, headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch(
  "https://PEM-SERVER-IP/pem/api/v17/server/{sid}/database/{db_name}/schema/{schema_
↪name}/function/{function_name}",
  {
    method: "GET",

    headers: headers,
  },
)
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

```
GET /server/{sid}/database/{db_name}/schema/{schema_name}/function/{function_name}
```

管理対象サーバーの特定のデータベースのスキーマにあるファンクションのプロパティを取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| sid | path | integer | true | サーバー ID |
| db_name | path | string | true | データベース名 |
| schema_name | path | string | true | スキーマ名 |
| function_name | path | string | true | なし |
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例

```
{
  "function_name": "string",
  "function_type": "string",
  "return_type": "string",
  "arg_types": ["string"],
  "function_binary": "string",
  "extension_name": "string",
  "package_name": "string",
  "schema_name": "string",
  "database_name": "string",
  "server_id": 0,
  "recorded_time": "2019-08-24T14:15:22Z"
}
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ステータス 200 | オペレーティングシステムの権限の制限によって影響を受ける機能 |

**注釈**

この操作には認証が必要ありません

## Profiles

Manage Profiles API ドラフトを作成、リスト、更新、公開/ロールバック、およびサーバーとエージェントのプロファイルの再割り当て/削除を行います。

### get_*profiles*

#### コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/profiles/ \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/profiles/, headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch("https://PEM-SERVER-IP/pem/api/v17/profiles/", {
  method: "GET",

  headers: headers,
})
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

```
GET /profiles/
```

#### リストプロファイル

公開されたプロファイルのみの概要リストを返します。公開されたプロファイルの下書きが存在する場合、項目にはそのプロファイルの ID が示され、’published’ は false です。応答には、’target_probe_configs’ または’target_alert_configs’ は含まれません。name 完全一致または name_contains 部分文字列検索クエリパラメーターを介してオプションの名前ベースのフィルタリングをサポートします。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| X-Auth-Token | header | string | true | 認証用のトークン |
| name | query | string | false | 名前の完全な一致大文字と小文字を区別してプロファイルをフィルタリングします。name_contains と相互排他的。 |
| name_contains | query | string | false | 部分文字列一致大文字と小文字を区別してプロファイルをフィルタリングします。 name と相互排他的。 |

### 応答例

```
[
  {
    "id": 34,
    "name": "Production Agents",
    "description": "Monitoring production agents",
    "target_kind": "a",
    "active_assignments": 0,
    "published": true
  }
]
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | Inline |

次のページに続く

表 130 – 前のページからの続き

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 400 | Bad Request | 相互排他性違反 `name`または類似の完全一致フィルタと`name_contains`または類似の部分文字列フィルタが両方とも同じ要求で指定されました。フィルタリングは、ペアのうちの 1 つのみを受け入れます。 | Inline |

### レスポンススキーマ

ステータスコード **200**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| _anonymous_ | *Platform Compatibility* ] | false | none | [リスト操作のトリミングされたプロファイル表現形式プローブ/アラート構成配列なし。 |
| » id | integer | false | none | なし |
| » name | string | false | none | なし |
| » description | string | false | none | なし |
| » target_kind | string | false | none | プロファイルのターゲットタイプサーバーの場合は's'、エージェントの場合は'a'。 |
| » published | boolean | false | none | コンビニエンスフラグステータスが'published'の場合に true。 |
| » active_assignments | integer | false | none | このプロファイルに割り当てられたアクティブなサーバー/エージェントの数。 |

### 列挙値

| Property | Value |
|---|---|
| target_kind | s |
| target_kind | a |

ステータスコード **400**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » success | integer | false | none | なし |
| » errormsg | string | false | none | なし |

**注釈**

この操作には認証が必要ありません

## post_profiles

### コードサンプル

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v17/profiles/ \
  -H Content-Type: application/json \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Content-Type: application/json,
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.post(https://PEM-SERVER-IP/pem/api/v17/profiles/, headers = headers)

print(r.json())
```

```
const inputBody = {
  "name": "Production Servers",
  "description": "Monitoring production database servers",
  "target_kind": "s"
};
```

(次のページに続く)

(前のページからの続き)

```
const headers = {
  Content-Type:application/json,
  Accept:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v17/profiles/,
{
  method: POST,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

POST /profiles/

### 新しいプロフィールを作成する

指定された名前必須、説明オプショナル、および target_kind オプショナル、サーバーのデフォルト's'を使用して新しいプロファイルを作成します。プローブと自動作成アラートは、target_kind に基づいて自動的に作成されます。

### ボディパラメーター

```
{
  "name": "Production Servers",
  "description": "Monitoring production database servers",
  "target_kind": "s"
}
```

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| X-Auth-Token | header | string | true | 承認用トークン |
| body | body | object | false | なし |
| » name | body | string | true | プロフィール名 必須 |
| » description | body | string | false | プロファイルの説明オプショナル |

次のページに続く

表 134 – 前のページからの続き

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| » target_kind | body | string | false | ターゲット種類 サーバーの場合は's'、エージェントの場合は'a' オプショナル、デフォルトは's' |

### 列挙値

| Parameter | Value |
|---|---|
| » target_kind | s |
| » target_kind | a |

### 応答例

```
{
  "success": 0,
  "data": "string"
}
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | プロファイルが正常に作成されました | Inline |
| 400 | Bad Request | 悪い要求-必須フィールド'name'が欠落しています | None |
| 500 | Internal Server Error | 内部サーバーエラー | None |

### レスポンススキーマ

ステータスコード **200**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » success | integer | false | none | なし |
| » data | string | false | none | なし |

> **注釈**
>
> この操作には認証が必要ありません

### get_*profiles*{profile_id}

#### コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/profiles/{profile_id} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/profiles/{profile_id}, headers =␣
↪headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch("https://PEM-SERVER-IP/pem/api/v17/profiles/{profile_id}", {
  method: "GET",

  headers: headers,
})
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

```
GET /profiles/{profile_id}
```

#### *ID* でプロフィールを取得

完全なプローブとアラート構成を含む単一のプロファイルを結果ます。応答には、ブール値の’published’

フィールドプロフィールが公開された場合に true が含まれます。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| profile_id | path | integer | true | なし |
| X-Auth-Token | header | string | true | 承認用トークン |

### 応答例


```
{
  "id": 0,
  "name": "string",
  "description": "string",
  "target_kind": "s",
  "published": true,
  "parent_id": 0,
  "draft_id": 0,
  "target_probe_configs": [
    {
      "probe_id": 0,
      "probe_name": "string",
      "enabled": true,
      "interval": 0,
      "lifetime": 0,
      "default_enabled": true,
      "default_interval": 0,
      "default_lifetime": 0,
      "target_type": 0,
      "force_enabled": true
    }
  ],
  "target_alert_configs": [
    {
      "id": 0,
      "alert_name": "string",
      "alert_template": 0,
      "description": "string",
      "enabled": true,
      "history_retention": 0,
      "frequency_min": 0,
      "auto_created": true,
      "template_id": "string",
      "params": [],
      "param_names": ["string"],
```


(次のページに続く)

(前のページからの続き)


```
      "param_types": "string",
      "params_units": ["string"],
      "threshold_unit": "string",
      "operator": "string",
      "low_threshold_value": 0,
      "medium_threshold_value": 0,
      "high_threshold_value": 0,
      "send_email": true,
      "all_alert_enable": true,
      "email_group_id": 0,
      "low_alert_enable": true,
      "low_email_group_id": 0,
      "med_alert_enable": true,
      "med_email_group_id": 0,
      "high_alert_enable": true,
      "high_email_group_id": 0,
      "send_trap": true,
      "snmp_trap_version": 0,
      "low_send_trap": true,
      "med_send_trap": true,
      "high_send_trap": true,
      "cleared_alert_enable": true,
      "execute_script": true,
      "execute_script_on_clear": true,
      "execute_script_on_pem_server": true,
      "script_code": "string",
      "submit_to_nagios": true,
      "override_default_config": true,
      "send_notification": true,
      "low_webhook_ids": [0],
      "med_webhook_ids": [0],
      "high_webhook_ids": [0],
      "cleared_webhook_ids": [0]
    }
  ]
}
```


プロファイルが見つかりません


```
{
  "message": "Not found"
}
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | *Upgrading SQL Profiler* |
| 404 | Not Found | プロファイルが見つかりません | Inline |

### レスポンススキーマ

ステータスコード **404**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » message | string | false | none | なし |

**注釈**

この操作には認証が必要ありません

### put_*profiles*{profile_id}

#### コードサンプル

```
#  You can also use wget

curl -X PUT https://PEM-SERVER-IP/pem/api/v17/profiles/{profile_id} \
  -H Content-Type: application/json \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Content-Type: application/json,
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.put(https://PEM-SERVER-IP/pem/api/v17/profiles/{profile_id}, headers =␣
↪headers)

print(r.json())
```

```
const inputBody = {
  "name": "Production Servers (v2)",
  "description": "Tightened thresholds and probe intervals.",
  "target_probe_configs": {
    "changed": [
      {
        "probe_id": 7,
        "execution_frequency": 5,
        "lifetime": 60
      },
      {
        "probe_id": 15,
        "enabled": true,
        "execution_frequency": 10,
        "lifetime": 30
      }
    ]
  },
  "target_alert_configs": {
    "changed": [
      {
        "id": 101,
        "enabled": true,
        "frequency_min": 2
      },
      {
        "id": 102,
        "operator": ">",
        "low_threshold_value": 70,
        "medium_threshold_value": 85,
        "high_threshold_value": 95
      }
    ],
    "added": [
      {
        "alert_name": "Replication Lag",
        "alert_template": 18,
        "enabled": true,
        "history_retention": 30,
        "frequency_min": 5,
        "operator": ">",
        "medium_threshold_value": 60,
        "high_threshold_value": 120
      }
    ],
```

(次のページに続く)

(前のページからの続き)

```
    "deleted": [
      {
        "id": 103
      }
    ]
  }
};
const headers = {
  Content-Type:application/json,
  Accept:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v17/profiles/{profile_id},
{
  method: PUT,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

PUT /profiles/{profile_id}

### *ID* でプロフィールを更新する

'profile_id'で指定される単一のプロファイルを更新します。メタデータには'name'/'description'を使用します。プローブの場合、'target_probe_configs.changed'を使用して、既存のプローブ構成のみを変更しますキーは probe_id、enabled?、enabled_by_default?、execute_frequency?、lifetime?。従来の'use_default_enabled'および'interval'も受け入れます。このエンドポイントを介したプローブの追加はサポートされていません。アラートの場合、変更/追加/削除とともに'target_alert_configs'を使用します削除は、'id'のみを持つオブジェクトを受け入れます。

### ボディパラメーター

```
{
  "name": "Production Servers (v2)",
  "description": "Tightened thresholds and probe intervals.",
  "target_probe_configs": {
    "changed": [
      {
        "probe_id": 7,
        "execution_frequency": 5,
```

(次のページに続く)

(前のページからの続き)


```
        "lifetime": 60
      },
      {
        "probe_id": 15,
        "enabled": true,
        "execution_frequency": 10,
        "lifetime": 30
      }
    ]
  },
  "target_alert_configs": {
    "changed": [
      {
        "id": 101,
        "enabled": true,
        "frequency_min": 2
      },
      {
        "id": 102,
        "operator": ">",
        "low_threshold_value": 70,
        "medium_threshold_value": 85,
        "high_threshold_value": 95
      }
    ],
    "added": [
      {
        "alert_name": "Replication Lag",
        "alert_template": 18,
        "enabled": true,
        "history_retention": 30,
        "frequency_min": 5,
        "operator": ">",
        "medium_threshold_value": 60,
        "high_threshold_value": 120
      }
    ],
    "deleted": [
      {
        "id": 103
      }
    ]
  }
}
```

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| profile_id | path | integer | true | なし |
| X-Auth-Token | header | string | true | 承認用トークン |
| body | body | object | true | なし |
| » name | body | string | false | プロファイルの新しい名前オプショナル。 |
| » description | body | string | false | プロファイルの新しい説明/注意事項オプショナル。 |
| » target_probe_configs | body | *ProbeConfigsUpdateOnly* | false | プロファイル ID で更新する場合のプローブ構成のデルタ 'changed' を使用して既存のプローブのみを変更します。このエンドポイントを介したプローブの追加はサポートされていません。 |
| » changed | body | *Platform Compatibility* ] | false | 変更する既存のプローブ構成 probe_id で識別されます。 |
| » probe_id | body | integer | true | 追加/更新するプローブ識別子。 |
| » enabled | body | boolean | false | プロファイルでこのプローブを有効/無効にします。 |
| » enabled_by_default | body | boolean | false | プローブがデフォルトの有効化に従うかどうか。エイリアス 'use_default_enabled' が受け入れられます。 |
| » execution_frequency | body | integer | false | 分単位のプローブ実行頻度。エイリアス'interval' が受け入れられます。 |
| » lifetime | body | integer | false | 履歴の保持日数。 |
| » target_alert_configs | body | *AlertConfigsDelta* | false | プロファイル内のアラート構成のデルタ。 |
| » changed | body | [allOf] | false | 既存のアラートを変更します。 |
| » id | body | integer | false | 変更するアラート ID。 |
| » name | body | string | false | アラート名。 |
| » enabled | body | boolean | false | なし |
| » history_retention | body | integer | false | なし |
| » frequency_min | body | integer | false | なし |

次のページに続く

表 141 – 前のページからの続き

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| » operator | body | string | false | なし |
| » low_threshold_value | body | number | false | なし |
| » » » medium_threshold_value | body | number | false | なし |
| » » » high_threshold_value | body | number | false | なし |
| » all_alert_enable | body | boolean | false | なし |
| » email_group_id | body | integer | false | なし |
| » low_alert_enable | body | boolean | false | なし |
| » low_email_group_id | body | integer | false | なし |
| » med_alert_enable | body | boolean | false | なし |
| » med_email_group_id | body | integer | false | なし |
| » high_alert_enable | body | boolean | false | なし |
| » » » high_email_group_id | body | integer | false | なし |
| » added | body | *Platform Compatibility* ] | false | 新しいアラートを追加します。必須フィールドについては、AlertPayload を参照してください。 |
| » alert_name | body | string | false | アラートの名前。注 このプロパティは、POST 要求の必須パラメーターです。 |
| » alert_template | body | integer | false | アラートテンプレートの ID。 ID は正の整数値である必要があります。注 このプロパティは、POST 要求の必須パラメーターです。 |
| » description | body | string | false | なし |
| » enabled | body | boolean | false | このアラートを有効/無効にするかどうかを定義します。注 このプロパティは、POST 要求の必須パラメーターです。 |
| » history_retention | body | integer | false | このアラートの履歴を保持する日数を指定します。指定できる値は 1 ～99999 です。 注このプロパティは、POST 要求の必須パラメーターです。 |

次のページに続く

表 141 – 前のページからの続き

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| » frequency_min | body | integer | false | PEM データベースサーバーでアラートを確認する頻度を分単位で定義します。指定できる値は 1〜65534 です。 注このプロパティは、POST 要求の必須パラメーターです。 |
| » operator | body | string | false | しきい値をチェックするための演算子を定義します。有効な文字列は>"および"<"です。注このプロパティは、POST 要求の必須パラメーターです。 |
| » low_threshold_value | body | number | false | 低アラートをトリガーするためのしきい値。このプロパティには、有効な整数/浮動小数点値を指定できます。注これは POST 要求の必須パラメーターです。 |
| » » » medium_threshold_value | body | number | false | 中アラートをトリガーするためのしきい値。このプロパティには、有効な整数/浮動小数点値を指定できます。注これは POST 要求の必須パラメーターです。 |
| » » » high_threshold_value | body | number | false | 高度なアラートをトリガーするためのしきい値。このプロパティには、有効な整数/浮動小数点値を指定できます。注これは POST 要求の必須パラメーターです。 |
| » all_alert_enable | body | boolean | false | アラートの状態が変更されたときに、パラメーター 'email_group_id' で指定されたグループにメール通知を送信するかどうかを決定します。 |

次のページに続く

表 141 – 前のページからの続き

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| » email_group_id | body | integer | false | アラートの状態が変更されたときに、メール通知の送信先のメールグループを定義します。 |
| » low_alert_enable | body | boolean | false | アラートの状態が LOW に設定されているときに、パラメーター 'low_email_group_id' で指定されたグループにメール通知を送信するかどうかを決定します。 |
| » low_email_group_id | body | integer | false | アラートの状態が LOW に変更されたときにメール通知の送信先の電子メールグループを定義します。 |
| » med_alert_enable | body | boolean | false | アラートの状態が MEDIUM に設定されている場合、パラメーター 'medium_email_group_id' で指定されたグループにメール通知を送信するかどうかを決定します。 |
| » med_email_group_id | body | integer | false | アラートの状態が MEDIUM に変更されたときにメール通知の送信先のメールグループを定義します。 |
| » high_alert_enable | body | boolean | false | アラートの状態が HIGH に設定されているときに、パラメーター 'high_email_group_id' で指定されたグループに電子メール通知を送信するかどうかを決定します。 |
| » » » high_email_group_id | body | integer | false | アラートの状態が HIGH に変更されたときに電子メール通知の送信先の電子メールグループを定義します。 |

次のページに続く

表 141 – 前のページからの続き

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| » send_trap | body | boolean | false | アラートの状態が変更されたときに SNMP トラップ通知を送信するかどうかを決定します。 |
| » snmp_trap_version | body | integer | false | SNMP トラップバージョンを定義します。サポートされている SNMP トラップバージョンは 1、2、または 3 です。 |
| » low_send_trap | body | boolean | false | アラートの状態が LOW に変更されたときに SNMP トラップ通知を送信するかどうかを決定します。 |
| » med_send_trap | body | boolean | false | アラートの状態が MEDIUM に変更されたときに SNMP トラップ通知を送信するかどうかを決定します。 |
| » high_send_trap | body | boolean | false | アラートの状態が HIGH に変化したときに SNMP トラップ通知を送信するかどうかを決定します。 |
| » cleared_alert_enable | body | boolean | false | 'クリアアラート' の通知をミュートするかどうかを決定します。 |
| » params | body | *Platform Compatibility* ] | false | アラートを確認するためにアラートテンプレートが使用するパラメーター/値のペアのリスト。注アラートテンプレートには、all パラメーターを定義する必要があります。 |
| » paramname | body | string | false | パラメーターの名前。注このプロパティは、アラートテンプレートパラメーターリストに存在する必要があります。例 |
| » paramvalue | body | string | false | 指定されたパラメーターの値 |

次のページに続く

表 141 – 前のページからの続き

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| » execute_script | body | boolean | false | アラートの状態が LOW/MEDIUM/HIGH に変更された場合に、提供されたスクリプトを実行するかどうかを決定します。 |
| » » » execute_script_on_clear | body | boolean | false | アラートの状態がクリアされた場合、提供されたスクリプトを実行するかどうかを決定します。 |
| » » » execute_script_on_pem_serve | body | boolean | false | 提供されたスクリプトを PEM ホストまたはエージェントがオブジェクトを監視しているホストで実行するかどうかを決定します。 |
| » script_code | body | string | false | 実行するシェルまたはバッチスクリプト。詳細については、ドキュメントをご覧ください。 |
| » submit_to_nagios | body | boolean | false | アラートの状態が変更されたときに、nagios に通知を送信するかどうかを定義します。 |
| » » » override_default_config | body | boolean | false | デフォルトの Webhook 構成をオーバーライドするかどうかを決定します。 |
| » send_notification | body | boolean | false | Webhook 送信通知を有効にするかどうかを決定します。 |
| » low_webhook_ids | body | [integer] | false | 低アラートを構成できる Webhook ID を定義します。 |
| » med_webhook_ids | body | [integer] | false | 中程度のアラートを構成できる Webhook ID を定義します。 |
| » high_webhook_ids | body | [integer] | false | 高アラートを構成できる Webhook ID を定義します。 |
| » » » cleared_webhook_ids | body | [integer] | false | クリアされたアラートを構成できる Webhook ID を定義します。 |

次のページに続く

表 141 – 前のページからの続き

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| » deleted | body | *Platform Compatibility* ] | false | ID ごとに既存のアラートを削除します。 |
| » id | body | integer | true | 削除するアラート構成 ID。 |

### 列挙値

| Parameter | Value |
|---|---|
| » operator | > |
| » operator | < |
| » paramname | Exclude tables smaller than |
| » paramname | mount point |
| » paramname | Host IP Address |
| » paramname | Time interval to monitor |
| » paramname | Time since idle |
| » paramname | Per CPU utilization (K%) |

### 応答例

```
{
  "message": "Invalid JSON body"
}
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | プロファイルが正常に更新されました | None |
| 400 | Bad Request | 無効な JSON 本文 | None |

### レスポンススキーマ

**注釈**

この操作には認証が必要ありません

### delete_*profiles*{profile_id}

コードサンプル

```
#  You can also use wget

curl -X DELETE https://PEM-SERVER-IP/pem/api/v17/profiles/{profile_id} \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  X-Auth-Token: string
}

r = requests.delete(https://PEM-SERVER-IP/pem/api/v17/profiles/{profile_id}, headers␣
↪= headers)

print(r.json())
```

```
const headers = {
  "X-Auth-Token": "string",
};

fetch("https://PEM-SERVER-IP/pem/api/v17/profiles/{profile_id}", {
  method: "DELETE",

  headers: headers,
})
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

```
DELETE /profiles/{profile_id}
```

プロフィールを削除する

#### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| profile_id | path | integer | true | なし |
| X-Auth-Token | header | string | true | 承認用トークン |

応答例

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | プロファイルは正常に削除されました | None |

### レスポンススキーマ

> **注釈**
>
> この操作には認証が必要ありません

### get_*profiles*{profile_id}_assigned

#### コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/profiles/{profile_id}/assigned \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/profiles/{profile_id}/assigned,␣
↪headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch("https://PEM-SERVER-IP/pem/api/v17/profiles/{profile_id}/assigned", {
  method: "GET",

  headers: headers,
```

(次のページに続く)

(前のページからの続き)

```
})
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

`GET /profiles/{profile_id}/assigned`

**プロファイルに割り当てられたサーバーとエージェントを取得する**

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| profile_id | path | integer | true | なし |
| X-Auth-Token | header | string | true | 承認用トークン |

**応答例**

```
{
  "servers": ["string"],
  "agents": ["string"]
}
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | Inline |

### レスポンススキーマ

ステータスコード **200**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » servers | [string] | false | none | なし |
| » agents | [string] | false | none | なし |

> **注釈**
>
> この操作には認証が必要ありません

### get_*profiles*{profile_id}_draft

#### コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/profiles/{profile_id}/draft \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/profiles/{profile_id}/draft,␣
↪headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch("https://PEM-SERVER-IP/pem/api/v17/profiles/{profile_id}/draft", {
  method: "GET",

  headers: headers,
})
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

```
GET /profiles/{profile_id}/draft
```

#### プロフィール *ID* でドラフトを取得

完全なプローブとアラート構成を含む、指定された公開プロファイル ID に関連付けられているドラフトプ

ロファイルを結果ます。応答には、parent_id または draft_id フィールドは含まれません。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| profile_id | path | integer | true | なし |
| X-Auth-Token | header | string | true | 承認用トークン |

### 応答例


```
{
  "id": 0,
  "name": "string",
  "description": "string",
  "target_kind": "s",
  "published": true,
  "target_probe_configs": [
    {
      "probe_id": 0,
      "probe_name": "string",
      "enabled": true,
      "interval": 0,
      "lifetime": 0,
      "default_enabled": true,
      "default_interval": 0,
      "default_lifetime": 0,
      "target_type": 0,
      "force_enabled": true
    }
  ],
  "target_alert_configs": [
    {
      "id": 0,
      "alert_name": "string",
      "alert_template": 0,
      "description": "string",
      "enabled": true,
      "history_retention": 0,
      "frequency_min": 0,
      "auto_created": true,
      "template_id": "string",
      "params": [],
      "param_names": ["string"],
      "param_types": "string",
      "params_units": ["string"],
```


(次のページに続く)

(前のページからの続き)


```
      "threshold_unit": "string",
      "operator": "string",
      "low_threshold_value": 0,
      "medium_threshold_value": 0,
      "high_threshold_value": 0,
      "send_email": true,
      "all_alert_enable": true,
      "email_group_id": 0,
      "low_alert_enable": true,
      "low_email_group_id": 0,
      "med_alert_enable": true,
      "med_email_group_id": 0,
      "high_alert_enable": true,
      "high_email_group_id": 0,
      "send_trap": true,
      "snmp_trap_version": 0,
      "low_send_trap": true,
      "med_send_trap": true,
      "high_send_trap": true,
      "cleared_alert_enable": true,
      "execute_script": true,
      "execute_script_on_clear": true,
      "execute_script_on_pem_server": true,
      "script_code": "string",
      "submit_to_nagios": true,
      "override_default_config": true,
      "send_notification": true,
      "low_webhook_ids": [0],
      "med_webhook_ids": [0],
      "high_webhook_ids": [0],
      "cleared_webhook_ids": [0]
    }
  ]
}
```


プロファイルのドラフトが見つかりません


```
{
  "message": "Not found"
}
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | プロフィールドラフト |
| 404 | Not Found | プロファイルのドラフトが見つかりませんでした | Inline |

### レスポンススキーマ

ステータスコード **404**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » message | string | false | none | なし |

**注釈**

この操作には認証が必要ありません

### delete_*profiles*{profile_id}_rollback

#### コードサンプル

```
#  You can also use wget

curl -X DELETE https://PEM-SERVER-IP/pem/api/v17/profiles/{profile_id}/rollback \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  X-Auth-Token: string
}

r = requests.delete(https://PEM-SERVER-IP/pem/api/v17/profiles/{profile_id}/rollback,␣
↪headers = headers)

print(r.json())
```

```
const headers = {
  "X-Auth-Token": "string",
```

(次のページに続く)

(前のページからの続き)

```
};

fetch("https://PEM-SERVER-IP/pem/api/v17/profiles/{profile_id}/rollback", {
  method: "DELETE",

  headers: headers,
})
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

`DELETE /profiles/{profile_id}/rollback`

プロファイルのロールバックドラフト

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| profile_id | path | integer | true | なし |
| X-Auth-Token | header | string | true | 承認用トークン |

### 応答例

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 下書きが削除されたか、使用可能なドラフトがありません | None |

### レスポンススキーマ

**注釈**

この操作には認証が必要ありません

## post_*profiles*{profile_id}_publish

コードサンプル

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v17/profiles/{profile_id}/publish \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.post(https://PEM-SERVER-IP/pem/api/v17/profiles/{profile_id}/publish,␣
↪headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch("https://PEM-SERVER-IP/pem/api/v17/profiles/{profile_id}/publish", {
  method: "POST",

  headers: headers,
})
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

POST /profiles/{profile_id}/publish

### プロフィールを公開する

指定されたプロファイルが存在する場合、そのドラフトを公開します。ドラフトが存在しない場合、プロファイルが既に公開されていることを示すメッセージを返します。

#### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| profile_id | path | integer | true | なし |
| X-Auth-Token | header | string | true | 承認用トークン |

### 応答例

```
{
  "message": "Not found"
}
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | プロファイルは正常に公開されたか、既に公開されました | None |
| 404 | Not Found | プロファイルが見つかりません | Inline |

### レスポンススキーマ

ステータスコード **404**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » message | string | false | none | なし |

**注釈**

この操作には認証が必要ありません

## post_*profiles*{source_id}_clone

### コードサンプル

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v17/profiles/{source_id}/clone \
  -H Content-Type: application/json \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Content-Type: application/json,
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.post(https://PEM-SERVER-IP/pem/api/v17/profiles/{source_id}/clone,␣
↪headers = headers)

print(r.json())
```

```
const inputBody = "{}";
const headers = {
  "Content-Type": "application/json",
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch("https://PEM-SERVER-IP/pem/api/v17/profiles/{source_id}/clone", {
  method: "POST",
  body: inputBody,
  headers: headers,
})
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

`POST /profiles/{source_id}/clone`

**既存のプロファイルのクローンを作成する**

すべてのプローブとアラート構成をソースからコピーすることにより、新しい公開プロファイルを作成します。ソースは公開されたプロファイルである必要があります。ドラフトのクローン作成は 400 で拒否されます。クローンはソースから独立しています。parent_id は NULL、ステータスは「published」、アクティブな割り当てはありません。クローンは 1 つの PEM インスタンス内で発生するため、すべての通知チャネル ID 電子メールグループ、Webhook ID、SNMP トラップフィールドはそのままコピーされます。自動作成されたアラートはコピーされ、クローンに削除できません。'name' が省略されると、サーバーは 'Copy of ' を生成し、名前が公開されたプロファイル間で一意になるまで、' (2) '、' (3)'、…を追加します。

**ボディパラメーター**

```
{}
```

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| source_id | path | integer | true | クローンする公開されたプロファイルの ID。 |
| X-Auth-Token | header | string | true | 承認用トークン |
| body | body | object | false | なし |
| » name | body | string | false | オプショナル。クローンの名前。公開されたプロファイル間で一意である必要があります。そうでない場合、要求は 409 を返します。省略すると、サーバーは競合回避で'Copy of <source />'を生成します。 |
| » description | body | string | false | オプショナル。クローンの説明。省略すると、ソースの説明がコピーされます。明示的な空の文字列は説明をクリアします。 |

#### 応答例

```
{
  "success": 1,
  "data": {
    "id": 42,
    "name": "Copy of Production Servers",
    "target_kind": "s",
    "description": "Monitoring production database servers"
  }
}
```

不正な要求。ソースはドラフトであり、クローンはできません。指定された名前が空である、自動生成された名前が使い果たされた、または‘name’ / ’description’は文字列ではありません。

```
{
  "errormsg": "Cannot clone a draft profile. Publish or rollback the draft first."
}
```

ソースプロファイルが見つかりません。

```
{
  "message": "Profile 999 does not exist."
}
```

409 レスポンス

```
{
  "message": "string"
}
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | プロファイルのクローンが正常に作成されました。 | Inline |
| 400 | Bad Request | 悪い要求。ソースはドラフトであり、クローンはできません。指定された名前が空である、自動生成された名前が使い果たされた、または 'name' / 'description' は文字列ではありません。 | Inline |
| 404 | Not Found | ソースプロファイルが見つかりません。 | Inline |
| 409 | 既存の *mod_wsgi* との競合 | ユーザー指定の 'name' は、既存の公開されたプロファイルと競合します。 | Inline |
| 500 | Internal Server Error | 内部サーバーエラー。 | None |

### レスポンススキーマ

ステータスコード **200**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » success | integer | false | none | なし |
| » data | object | false | none | なし |
| » id | integer | false | none | 新しいプロファイルのID。 |
| » name | string | false | none | なし |
| » target_kind | string | false | none | ソースから保存。クローン時には変更できません。 |
| » description | string ¦ null | false | none | なし |

### 列挙値

| Property | Value |
|---|---|
| target_kind | s |
| target_kind | a |

ステータスコード **400**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » errormsg | string | false | none | なし |

ステータスコード **404**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » message | string | false | none | なし |

ステータスコード **409**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » message | string | false | none | なし |

**注釈**

この操作には認証が必要ありません

### delete_*profiles*{profile_id}_{new_profile_id}

コードサンプル

```
#  You can also use wget

curl -X DELETE https://PEM-SERVER-IP/pem/api/v17/profiles/{profile_id}/{new_profile_
↪id} \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  X-Auth-Token: string
}

r = requests.delete(https://PEM-SERVER-IP/pem/api/v17/profiles/{profile_id}/{new_
↪profile_id}, headers = headers)

print(r.json())
```

```
const headers = {
  "X-Auth-Token": "string",
};

fetch(
  "https://PEM-SERVER-IP/pem/api/v17/profiles/{profile_id}/{new_profile_id}",
  {
    method: "DELETE",

    headers: headers,
  },
)
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

`DELETE /profiles/{profile_id}/{new_profile_id}`

プロファイルを削除し、新しいプロファイルに再割り当てします

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| profile_id | path | integer | true | なし |
| new_profile_id | path | integer | true | なし |
| X-Auth-Token | header | string | true | 承認用トークン |

### 応答例

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | プロファイルは正常に削除および再割り当てされました | None |

### レスポンススキーマ

**注釈**

この操作には認証が必要ありません

## post_profile_export

### コードサンプル

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v17/profile/export/ \
  -H Content-Type: application/json \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Content-Type: application/json,
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.post(https://PEM-SERVER-IP/pem/api/v17/profile/export/, headers =␣
↪headers)
```

(次のページに続く)

(前のページからの続き)

```
print(r.json())
```

```
const inputBody = {
  "profiles": [
    0
  ]
};
const headers = {
  Content-Type:application/json,
  Accept:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v17/profile/export/,
{
  method: POST,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

POST /profile/export/

**プロファイルを *JSON* バンドルにエクスポートする**

.pemprf JSON バンドルとしてシリアル化された要求されたプロファイルを返します。ドラフトを選択すると、公開された親に透過的に解決されます。すべての電子メール/SNMP/webhook/Nagios チャネル構成はエクスポート時にストリップされます—ルーティング ID email_group_id 、*_email_group_id 、* _webhook_ids 、snmp_trap_version 、*_send_trap および有効化フラグ send_email 、send_trap 、submit_to_nagios 、cleared_alert_enable 、send_notification 、受信者 opt2 のイネーブルフラグ cleared_alert_enable 、send_notification 、edb_tran_s_opt2 インポート後の UI を介してチャンネルごとに。スクリプトの実行 execute_script 、execute_script_on_clear 、execute_script_on_pem_server 、および script_code 本体は保持されます。 script_code には資格情報または env 固有のパスが含まれる場合があるため、 .pemprf ファイルを機密ファイルとして扱います。

**ボディパラメーター**

```
{
  "profiles": [0]
}
```

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| X-Auth-Token | header | string | true | 承認用トークン |
| body | body | object | true | なし |
| » profiles | body | [integer] | true | エクスポートするプロファイル ID。 |

### 応答例

```
{
  "version": 0,
  "profiles": [{}]
}
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | バンドル application/json をエクスポートします。 | Inline |
| 400 | Bad Request | エクスポートするプロファイルがありません、プロファイル ID が見つかりません、孤立したドラフト、または解決されたプロファイルは公開されません。 | None |
| 401 | Unauthorized | ユーザーには `pem_manage_profile ルがありません。 | None |

### レスポンススキーマ

ステータスコード **200**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » version | integer | false | none | エクスポートファイル形式のバージョン。 |

次のページに続く

表 168 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » profiles | [object] | false | none | なし |

**注釈**

この操作には認証が必要ありません

### post_profile_import

#### コードサンプル

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v17/profile/import/ \
  -H Content-Type: application/json \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Content-Type: application/json,
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.post(https://PEM-SERVER-IP/pem/api/v17/profile/import/, headers =␣
↪headers)

print(r.json())
```

```
const inputBody = {
  "version": 0,
  "profiles": [
    {}
  ],
  "skip_overwrite": true
};
const headers = {
  Content-Type:application/json,
  Accept:application/json,
  X-Auth-Token:string
};
```

(次のページに続く)

(前のページからの続き)

```
fetch(https://PEM-SERVER-IP/pem/api/v17/profile/import/,
{
  method: POST,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

```
POST /profile/import/
```

#### 以前にエクスポートしたプロファイルをインポートする

ファイル内の各プロファイルは独立して処理されます。新しい名前は公開されたとおりに着陸します。名前の競合デフォルトの Skipped skip_overwrite=true か、既存のプロファイルの構成をドラフトとして置き換えます skip_overwrite=false —ユーザーは UI で [適用] をクリックして公開します。応答は、プロファイルごとに 1 つの結果を伝送します。status は Success 、 Skipped 、または Failed で、 msg は部分スキップの詳細プローブ/テンプレート参照の欠如を説明します。

#### ボディパラメーター

```
{
  "version": 0,
  "profiles": [{}],
  "skip_overwrite": true
}
```

#### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| X-Auth-Token | header | string | true | 承認用トークン |
| body | body | object | true | なし |
| » version | body | integer | true | エクスポートファイル形式のバージョン。ターゲットサーバーでサポートされている必要があります。 |
| » profiles | body | [object] | true | なし |

次のページに続く

表 169 – 前のページからの続き

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| » skip_overwrite | body | boolean | true | true の場合、名前の競合により `Skipped`が生成されます。 false の場合、インポーターは既存のプロファイルにドラフトを作成しますレビューと公開に適用します。 |

**応答例**

```
{
  "success": 0,
  "result": [
    {
      "name": "string",
      "status": "Success",
      "msg": "string"
    }
  ]
}
```

**応答**

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | プロファイルごとのインポート結果。 | Inline |
| 400 | Bad Request | 不正なペイロード、skip_overwrite の欠落、またはサポートされていないバージョン。 | None |
| 401 | Unauthorized | ユーザーには `pem_manage_profile ルがありません。 | None |

**レスポンススキーマ**

ステータスコード **200**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » success | integer | false | none | なし |
| » result | [object] | false | none | なし |
| » name | string | false | none | なし |
| » status | string | false | none | なし |
| » msg | string ¦ null | false | none | なし |

#### 列挙値

| Property | Value |
|---|---|
| status | Success |
| status | Skipped |
| status | Failed |

**注釈**

この操作には認証が必要ありません

### Blackout

アラートのブラックアウトを管理します。ブラックアウトは、定義された期間、特定のエージェントまたはサーバーのアラートを抑制します。

#### get_*blackout*

**コードサンプル**

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/blackout/ \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/blackout/, headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch("https://PEM-SERVER-IP/pem/api/v17/blackout/", {
  method: "GET",

  headers: headers,
})
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /blackout/

### リスト停電

pem.blackout からブラックアウトレコードを返しますアクティブ+将来のスケジュール済み。オプションで object_type、object_id、または source でフィルターします。include_history=true を渡して、pem.blackout_history からアーカイブされたレコードも結果ます。

#### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| X-Auth-Token | header | string | true | 認証用トークン |
| object_type | query | integer | false | オブジェクトタイプでフィルターします 100 エージェントまたは 200 サーバー |
| object_id | query | integer | false | 特定のオブジェクト ID でフィルタリング |
| source | query | string | false | ソースによるフィルタリング手動、スケジュール、または自動 |
| include_history | query | boolean | false | true の場合、アーカイブされたブラックアウトレコードも含めます |

#### 列挙値

| Parameter | Value |
|---|---|
| object_type | 100 |

次のページに続く

表 174 – 前のページからの続き

| Parameter | Value |
|---|---|
| object_type | 200 |
| source | manual |
| source | scheduled |
| source | auto |

### 応答例

```
{
  "blackouts": [
    {
      "id": 0,
      "start_time": "2019-08-24T14:15:22Z",
      "end_time": "2019-08-24T14:15:22Z",
      "object_type": 0,
      "object_id": 0,
      "source": "manual",
      "batch_id": "4da22c97-b7d5-4e31-8c3a-03870ebc7b20",
      "created_by": "string",
      "created_at": "2019-08-24T14:15:22Z"
    }
  ]
}
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ブラックアウトレコードのリスト | Inline |

### レスポンススキーマ

ステータスコード **200**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » blackouts | [oneOf] | false | none | アクティブなブラックアウトレコード。include_history=true の場合、追加の actual_end_time および archived フィールドを備えた BlackoutHistory オブジェクトも含まれる場合があります。 |

*oneOf*

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » _anonymous_ | アラートブラックアウト管理 | false | none | なし |
| » id | integer | false | none | なし |
| » start_time | string(date-time) | false | none | なし |
| » end_time | string(date-time) ¦ null | false | none | なし |
| » object_type | integer | false | none | 100=エージェント、200=サーバー |
| » object_id | integer | false | none | なし |
| » source | string | false | none | なし |
| » batch_id | string(uuid) ¦ null | false | none | なし |
| » created_by | string | false | none | なし |
| » created_at | string(date-time) | false | none | なし |

*xor*

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » _anonymous_ | *BlackoutHistory* | false | none | アーカイブされたブラックアウトレコード。 GET /blackout/?include_history=true ライブ行と混合、および ID が pem.blackout_history でのみ見つかった場合に GET /blackout/{id}によって返されます。 |
| » id | integer | false | none | なし |
| » start_time | string(date-time) | false | none | なし |
| » scheduled_end_time | string(date-time)¦ null | false | none | ブラックアウトが作成されたときの元の end_time |
| » actual_end_time | string(date-time) | false | none | 実際に停電が終了したとき |
| » object_type | integer | false | none | 100=エージェント、200=サーバー |
| » object_id | integer | false | none | なし |
| » source | string | false | none | なし |
| » batch_id | string(uuid)¦ null | false | none | なし |
| » created_by | string | false | none | なし |
| » created_at | string(date-time) | false | none | なし |
| » archived | boolean | false | none | 履歴レコードの場合は常に true |

### 列挙値

| Property | Value |
|---|---|
| source | manual |
| source | scheduled |
| source | auto |
| source | manual |
| source | scheduled |
| source | auto |

**注釈**

この操作には認証が必要ありません

### post_*blackout*

#### コードサンプル

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v17/blackout/ \
  -H Content-Type: application/json \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Content-Type: application/json,
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.post(https://PEM-SERVER-IP/pem/api/v17/blackout/, headers = headers)

print(r.json())
```

```
const inputBody = {
  "start_time": "2019-08-24T14:15:22Z",
  "duration": 1,
  "object_type": 100,
  "object_ids": [
    0
  ]
};
const headers = {
  Content-Type:application/json,
  Accept:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v17/blackout/,
{
  method: POST,
  body: inputBody,
  headers: headers
})
.then(function(res) {
```

(次のページに続く)

(前のページからの続き)

```
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

POST /blackout/

### *1* つ以上のブラックアウトを作成する

指定されたオブジェクトのブラックアウトレコードを作成します。バッチを作成するために複数の object_id を指定できます。

### ボディパラメーター

```
{
  "start_time": "2019-08-24T14:15:22Z",
  "duration": 1,
  "object_type": 100,
  "object_ids": [0]
}
```

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| X-Auth-Token | header | string | true | 認証用トークン |
| body | body | object | true | なし |
| » start_time | body | string(date-time) | true | ブラックアウトが始まるとき ISO 8601 |
| » duration | body | integer | true | 停電の時間単位。 1〜24 の整数であり、REST で作成されたブラックアウトが UI で正しくレンダリングされるように、[ Schedule Alert Blackout ] ダイアログが提供する値と一致する必要があります。 |
| » object_type | body | integer | true | 100 =エージェント、200 =サーバー |
| » object_ids | body | [integer] | true | エージェントまたはサーバー ID の配列 |

### 列挙値

| Parameter | Value |
|---|---|
| » duration | 1 |
| » duration | 2 |

次のページに続く

表 181 – 前のページからの続き

| Parameter | Value |
|---|---|
| » duration | 3 |
| » duration | 4 |
| » duration | 5 |
| » duration | 6 |
| » duration | 7 |
| » duration | 8 |
| » duration | 9 |
| » duration | 10 |
| » duration | 11 |
| » duration | 12 |
| » duration | 13 |
| » duration | 14 |
| » duration | 15 |
| » duration | 16 |
| » duration | 17 |
| » duration | 18 |
| » duration | 19 |
| » duration | 20 |
| » duration | 21 |
| » duration | 22 |
| » duration | 23 |
| » duration | 24 |
| » object_type | 100 |
| » object_type | 200 |

**応答例**

```
{
  "ids": [0],
  "batch_id": "4da22c97-b7d5-4e31-8c3a-03870ebc7b20"
}
```

**応答**

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 201 | *8.* **スケジュールされた停電が作成されましたダイアログ** | ブラックアウトが正常に作成されました | Inline |
| 428 | Precondition Required | 検証エラー 無効な入力、制約違反 | None |

### レスポンススキーマ

ステータスコード **201**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » ids | [integer] | false | none | なし |
| » batch_id | string(uuid) | false | none | なし |

**注釈**

この操作には認証が必要ありません

### get_*blackout*{blackout_id}

#### コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/blackout/{blackout_id} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/blackout/{blackout_id}, headers =␣
↪headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch("https://PEM-SERVER-IP/pem/api/v17/blackout/{blackout_id}", {
  method: "GET",

  headers: headers,
```

(次のページに続く)

(前のページからの続き)

```
})
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

```
GET /blackout/{blackout_id}
```

### 特定のブラックアウトを取得する

ブラックアウトレコードを返します。ID が pem.blackout に存在しない場合、ハンドラーは pem.blackout_history にフォールバックし、アーカイブされたレコードを結果ます”archived”:true および actual_end_time / stable_end_time フィールドの存在によって区別されます。ID がどのテーブルでも見つからない場合、404 を返します。

#### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| X-Auth-Token | header | string | true | 認証用トークン |
| blackout_id | path | integer | true | なし |

#### 応答例

```
{
  "id": 0,
  "start_time": "2019-08-24T14:15:22Z",
  "end_time": "2019-08-24T14:15:22Z",
  "object_type": 0,
  "object_id": 0,
  "source": "manual",
  "batch_id": "4da22c97-b7d5-4e31-8c3a-03870ebc7b20",
  "created_by": "string",
  "created_at": "2019-08-24T14:15:22Z"
}
```

#### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ブラックアウトレコード。pem.blackout からのライブ行 Blackout スキーマ、または id が pem.blackout_history でのみ見つかった場合、アーカイブされた行 archived=true の BlackoutHistory スキーマ。 | Inline |
| 404 | Not Found | ブラックアウトが見つかりません | None |

### レスポンススキーマ

### 列挙値

| Property | Value |
|---|---|
| source | manual |
| source | scheduled |
| source | auto |
| source | manual |
| source | scheduled |
| source | auto |

**注釈**

この操作には認証が必要ありません

### delete_*blackout*{blackout_id}

### コードサンプル

```
#  You can also use wget

curl -X DELETE https://PEM-SERVER-IP/pem/api/v17/blackout/{blackout_id} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.delete(https://PEM-SERVER-IP/pem/api/v17/blackout/{blackout_id}, headers␣
↪= headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch("https://PEM-SERVER-IP/pem/api/v17/blackout/{blackout_id}", {
  method: "DELETE",

  headers: headers,
})
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

```
DELETE /blackout/{blackout_id}
```

#### 停電の終了

ブラックアウトレコードを終了アーカイブし、履歴に移動します。

#### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| X-Auth-Token | header | string | true | 認証用トークン |
| blackout_id | path | integer | true | なし |

#### 応答例

```
{
  "id": 0,
  "deleted": true,
```

(次のページに続く)

(前のページからの続き)


```
  "warnings": [
    {
      "code": "string",
      "message": "string"
    }
  ]
}
```


### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ブラックアウトが終了およびアーカイブされた | Inline |
| 404 | Not Found | ブラックアウトが見つかりません | None |

### レスポンススキーマ

ステータスコード **200**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » id | integer | false | none | 終了したブラックアウトの ID |
| » deleted | boolean | false | none | 成功した場合は常に true |
| » warnings | *Platform Compatibility* ] | false | none | オプションの非致命的な警告 — アーカイブの INSERT が影響した行が、DELETE が返した行よりも少ない場合にのみ存在しますたとえば、pem.blackout_history の RLS またはトリガードリフト。各警告には、診断用の BlackoutId、ArchivedCount、および ExpectedArchived が含まれます。 |

次のページに続く

表 189 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » code | string | true | none | 安定した警告識別子。既知のコード *manual_post_state_unavailable* COMMIT 後にトグルレスポンス形状の SELECT が失敗した、*in_blackout_post_state_unavailable* 同期フラグの場合、*flag_select_failed* ステータスヘルパーはレコード派生の in_blackout にフォールバック、*history_archive_incomplete* DELETE は予想よりも少ない行をアーカイブしました-RLS /トリガーを調査します pem.blackout_history で)。 |
| » message | string | true | none | なし |

**注釈**

この操作には認証が必要ありません

### get_*blackout_server*{sid}

#### コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/blackout/server/{sid} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/blackout/server/{sid}, headers =␣
```

(次のページに続く)

(前のページからの続き)

```
↪headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch("https://PEM-SERVER-IP/pem/api/v17/blackout/server/{sid}", {
  method: "GET",

  headers: headers,
})
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /blackout/server/{sid}

### サーバーのブラックアウトステータスの取得

同期された alert_blackout フラグと、指定されたサーバーのソース別アクティブレコードの内訳を結果ます。

#### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| X-Auth-Token | header | string | true | 認証用トークン |
| sid | path | integer | true | サーバー ID |

#### 応答例

```
{
  "object_type": 100,
  "object_id": 0,
  "in_blackout": true,
  "manual": {
    "id": 0,
    "start_time": "2019-08-24T14:15:22Z",
    "end_time": "2019-08-24T14:15:22Z",
    "object_type": 0,
    "object_id": 0,
```

(次のページに続く)

(前のページからの続き)


```
    "source": "manual",
    "batch_id": "4da22c97-b7d5-4e31-8c3a-03870ebc7b20",
    "created_by": "string",
    "created_at": "2019-08-24T14:15:22Z"
  },
  "scheduled": [
    {
      "id": 0,
      "start_time": "2019-08-24T14:15:22Z",
      "end_time": "2019-08-24T14:15:22Z",
      "object_type": 0,
      "object_id": 0,
      "source": "manual",
      "batch_id": "4da22c97-b7d5-4e31-8c3a-03870ebc7b20",
      "created_by": "string",
      "created_at": "2019-08-24T14:15:22Z"
    }
  ],
  "auto": {
    "id": 0,
    "start_time": "2019-08-24T14:15:22Z",
    "end_time": "2019-08-24T14:15:22Z",
    "object_type": 0,
    "object_id": 0,
    "source": "manual",
    "batch_id": "4da22c97-b7d5-4e31-8c3a-03870ebc7b20",
    "created_by": "string",
    "created_at": "2019-08-24T14:15:22Z"
  },
  "warnings": [
    {
      "code": "string",
      "message": "string"
    }
  ]
}
```


応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | サーバーのブラックアウトステータス。 | *ObjectBlackoutStatus* |
| 404 | Not Found | 指定された ID のサーバーは存在しません。 | None |
| 428 | Precondition Required | サーバー ID がパスに指定されませんでした。 | None |

**注釈**

この操作には認証が必要ありません

### put_*blackout_server*{sid}

#### コードサンプル

```
#  You can also use wget

curl -X PUT https://PEM-SERVER-IP/pem/api/v17/blackout/server/{sid} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.put(https://PEM-SERVER-IP/pem/api/v17/blackout/server/{sid}, headers =␣
↪headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch("https://PEM-SERVER-IP/pem/api/v17/blackout/server/{sid}", {
  method: "PUT",
```

(次のページに続く)

(前のページからの続き)

```
  headers: headers,
})
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

PUT /blackout/server/{sid}

### サーバーの手動ブラックアウトを有効にする

冪等。このサーバーに存在しない場合は、手動の不定レコードを挿入します。それ以外の場合は、Changed=false を使用して既存のレコードを結果ます。オブジェクトごとに 1 つのアクティブな手動レコードのみが許可されます pem.idx_blackout_one_active_in individually によって適用されます。

#### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| X-Auth-Token | header | string | true | 認証用トークン |
| sid | path | integer | true | サーバー ID |

#### 応答例

```
{
  "changed": true,
  "in_blackout": true,
  "manual": {
    "id": 0,
    "start_time": "2019-08-24T14:15:22Z",
    "end_time": "2019-08-24T14:15:22Z",
    "object_type": 0,
    "object_id": 0,
    "source": "manual",
    "batch_id": "4da22c97-b7d5-4e31-8c3a-03870ebc7b20",
    "created_by": "string",
    "created_at": "2019-08-24T14:15:22Z"
  },
  "ended_records": [
    {
      "id": 0,
      "source": "manual",
      "scheduled_end": "2019-08-24T14:15:22Z",
```

(次のページに続く)

(前のページからの続き)


```
      "ended_early": true
    }
  ],
  "warnings": [
    {
      "code": "string",
      "message": "string"
    }
  ]
}
```


### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | トグル結果。最初に有効にしたときに Changed=true。べき等の no-op の Changed=false。 | *ObjectBlackoutToggleResult* |
| 404 | Not Found | 指定された ID のオブジェクトは存在しません。 | None |

**注釈**

この操作には認証が必要ありません

## delete_*blackout_server*{sid}

### コードサンプル

```
#  You can also use wget

curl -X DELETE https://PEM-SERVER-IP/pem/api/v17/blackout/server/{sid} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}
```

(次のページに続く)

(前のページからの続き)

```
r = requests.delete(https://PEM-SERVER-IP/pem/api/v17/blackout/server/{sid}, headers=
↪= headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch("https://PEM-SERVER-IP/pem/api/v17/blackout/server/{sid}", {
  method: "DELETE",

  headers: headers,
})
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

```
DELETE /blackout/server/{sid}
```

### サーバー上のすべての停電を終了する

冪等。ソース手動+スケジュール+自動に関係なく、このサーバー上のすべてのアクティブなブラックアウトレコードを終了します。ダッシュボードのチェックボックスの既にアクティブな動作のチェックを解除すると一致します。ユーザーのインテントは「アラートをオンにします」です。開始されたレコードは pem.blackout_history にアーカイブされます。将来にスケジュールされたレコードは単に削除されます。

#### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| X-Auth-Token | header | string | true | 認証用トークン |
| sid | path | integer | true | サーバー ID |

#### 応答例

```
{
  "changed": true,
  "in_blackout": true,
  "manual": {
    "id": 0,
```

(次のページに続く)

(前のページからの続き)


```
    "start_time": "2019-08-24T14:15:22Z",
    "end_time": "2019-08-24T14:15:22Z",
    "object_type": 0,
    "object_id": 0,
    "source": "manual",
    "batch_id": "4da22c97-b7d5-4e31-8c3a-03870ebc7b20",
    "created_by": "string",
    "created_at": "2019-08-24T14:15:22Z"
  },
  "ended_records": [
    {
      "id": 0,
      "source": "manual",
      "scheduled_end": "2019-08-24T14:15:22Z",
      "ended_early": true
    }
  ],
  "warnings": [
    {
      "code": "string",
      "message": "string"
    }
  ]
}
```


### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | トグル結果。少なくとも 1 つのレコードが終了した場合、Changed=true。べき等の no-op の Changed=false。 | *ObjectBlackoutToggleResult* |
| 404 | Not Found | 指定された ID のオブジェクトは存在しません。 | None |

**注釈**

この操作には認証が必要ありません

### get_*blackout_agent*{aid}

#### コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/blackout/agent/{aid} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/blackout/agent/{aid}, headers =␣
↪headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch("https://PEM-SERVER-IP/pem/api/v17/blackout/agent/{aid}", {
  method: "GET",

  headers: headers,
})
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

`GET /blackout/agent/{aid}`

#### エージェントのブラックアウトステータスの取得

同期された alert_blackout フラグと、指定されたエージェントのソース別アクティブレコードの内訳を返します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| X-Auth-Token | header | string | true | 認証用トークン |
| aid | path | integer | true | エージェント ID |

### 応答例


```
{
  "object_type": 100,
  "object_id": 0,
  "in_blackout": true,
  "manual": {
    "id": 0,
    "start_time": "2019-08-24T14:15:22Z",
    "end_time": "2019-08-24T14:15:22Z",
    "object_type": 0,
    "object_id": 0,
    "source": "manual",
    "batch_id": "4da22c97-b7d5-4e31-8c3a-03870ebc7b20",
    "created_by": "string",
    "created_at": "2019-08-24T14:15:22Z"
  },
  "scheduled": [
    {
      "id": 0,
      "start_time": "2019-08-24T14:15:22Z",
      "end_time": "2019-08-24T14:15:22Z",
      "object_type": 0,
      "object_id": 0,
      "source": "manual",
      "batch_id": "4da22c97-b7d5-4e31-8c3a-03870ebc7b20",
      "created_by": "string",
      "created_at": "2019-08-24T14:15:22Z"
    }
  ],
  "auto": {
    "id": 0,
    "start_time": "2019-08-24T14:15:22Z",
    "end_time": "2019-08-24T14:15:22Z",
    "object_type": 0,
    "object_id": 0,
    "source": "manual",
    "batch_id": "4da22c97-b7d5-4e31-8c3a-03870ebc7b20",
    "created_by": "string",
```


(次のページに続く)

(前のページからの続き)


```
    "created_at": "2019-08-24T14:15:22Z"
  },
  "warnings": [
    {
      "code": "string",
      "message": "string"
    }
  ]
}
```


### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | エージェントのブラックアウトステータス。 | *ObjectBlackoutStatus* |
| 404 | Not Found | 指定された ID のエージェントは存在しません。 | None |
| 428 | Precondition Required | パスにエージェント ID が指定されませんでした。 | None |

**注釈**

この操作には認証が必要ありません

### put_*blackout_agent*{aid}

#### コードサンプル

```
#  You can also use wget

curl -X PUT https://PEM-SERVER-IP/pem/api/v17/blackout/agent/{aid} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
```

(次のページに続く)

(前のページからの続き)

```
}

r = requests.put(https://PEM-SERVER-IP/pem/api/v17/blackout/agent/{aid}, headers =␣
↪headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch("https://PEM-SERVER-IP/pem/api/v17/blackout/agent/{aid}", {
  method: "PUT",

  headers: headers,
})
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

PUT /blackout/agent/{aid}

### エージェントの手動ブラックアウトを有効にする

冪等。このエージェントに存在しない場合、手動の不定レコードを挿入します。それ以外の場合は、Changed=false を使用して既存のレコードを結果ます。オブジェクトごとに 1 つのアクティブな手動レコードのみが許可されます pem.idx_blackout_one_active_in individually によって適用されます。

#### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| X-Auth-Token | header | string | true | 認証用トークン |
| aid | path | integer | true | エージェント ID |

### 応答例

```
{
  "changed": true,
  "in_blackout": true,
  "manual": {
    "id": 0,
```

(次のページに続く)

(前のページからの続き)


```
    "start_time": "2019-08-24T14:15:22Z",
    "end_time": "2019-08-24T14:15:22Z",
    "object_type": 0,
    "object_id": 0,
    "source": "manual",
    "batch_id": "4da22c97-b7d5-4e31-8c3a-03870ebc7b20",
    "created_by": "string",
    "created_at": "2019-08-24T14:15:22Z"
  },
  "ended_records": [
    {
      "id": 0,
      "source": "manual",
      "scheduled_end": "2019-08-24T14:15:22Z",
      "ended_early": true
    }
  ],
  "warnings": [
    {
      "code": "string",
      "message": "string"
    }
  ]
}
```


### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | トグル結果。最初に有効にしたときに Changed=true。べき等の no-op の Changed=false。 | *ObjectBlackoutToggleResult* |
| 404 | Not Found | 指定された ID のオブジェクトは存在しません。 | None |

**注釈**

この操作には認証が必要ありません

### delete_*blackout_agent*{aid}

#### コードサンプル

```
#  You can also use wget

curl -X DELETE https://PEM-SERVER-IP/pem/api/v17/blackout/agent/{aid} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.delete(https://PEM-SERVER-IP/pem/api/v17/blackout/agent/{aid}, headers =␣
↪headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch("https://PEM-SERVER-IP/pem/api/v17/blackout/agent/{aid}", {
  method: "DELETE",

  headers: headers,
})
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

```
DELETE /blackout/agent/{aid}
```

#### エージェントのすべてのブラックアウトを終了する

冪等。ソース手動+スケジュール+自動に関係なく、このエージェントのすべてのアクティブなブラックアウトレコードを終了します。ダッシュボードのチェックボックスの既にアクティブな動作のチェックを解除すると一致します。ユーザーのインテントは「アラートをオンにします」です。開始されたレコードは pem.blackout_history にアーカイブされます。将来にスケジュールされたレコードは単に削除されます。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| X-Auth-Token | header | string | true | 認証用トークン |
| aid | path | integer | true | エージェント ID |

### 応答例


```
{
  "changed": true,
  "in_blackout": true,
  "manual": {
    "id": 0,
    "start_time": "2019-08-24T14:15:22Z",
    "end_time": "2019-08-24T14:15:22Z",
    "object_type": 0,
    "object_id": 0,
    "source": "manual",
    "batch_id": "4da22c97-b7d5-4e31-8c3a-03870ebc7b20",
    "created_by": "string",
    "created_at": "2019-08-24T14:15:22Z"
  },
  "ended_records": [
    {
      "id": 0,
      "source": "manual",
      "scheduled_end": "2019-08-24T14:15:22Z",
      "ended_early": true
    }
  ],
  "warnings": [
    {
      "code": "string",
      "message": "string"
    }
  ]
}
```


### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | トグル結果。少なくとも 1 つのレコードが終了した場合、Changed=true。べき等の no-op の Changed=false。 | *ObjectBlackoutToggleResult* |
| 404 | Not Found | 指定された ID のオブジェクトは存在しません。 | None |

**注釈**

この操作には認証が必要ありません

### Monitored Object Status

#### get_*agent_status*

コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/agent/status/ \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/agent/status/, headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch("https://PEM-SERVER-IP/pem/api/v17/agent/status/", {
```

(次のページに続く)

(前のページからの続き)

```
  method: "GET",

  headers: headers,
})
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /agent/status/

**表示されているすべての *pem* エージェントのステータスを取得します。**

ユーザーに表示される pem エージェントの情報とステータスのリストを返しますトークンで表されます。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| X-Auth-Token | header | string | true | 認証用のトークン |

**応答例**

```
[
  {
    "id": 0,
    "name": "string",
    "group_id": 0,
    "group_name": "string",
    "blackout": true,
    "status": "UP",
    "alerts": {
      "total": 0,
      "acknowledged": 0,
      "high": 0,
      "medium": 0,
      "low": 0,
      "high_acknowledged": 0,
      "medium_acknowledged": 0,
      "low_acknowledged": 0
    },
    "version": "string",
    "processes": 0,
    "threads": 0,
```

(次のページに続く)

(前のページからの続き)


```
    "cpu_utilization": 0,
    "memory_utilization": 0,
    "swap_utilization": 0,
    "disk_utilization": 0
  }
]
```


### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | pem エージェントに関する情報とステータスのリストを返します。 | Inline |

### レスポンススキーマ

ステータスコード **200**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| _anonymous_ | *Platform Compatibility* ] | false | none | なし |
| » id | integer | false | none | エージェント ID |
| » name | string | false | none | エージェント名 |
| » group_id | integer | false | none | エージェントがユーザーに表示されるグループ ID |
| » group_name | string | false | none | グループ名 |
| » blackout | boolean | false | none | エージェントのアラートブラックアウトを有効にするか無効にするかを決定します。 |
| » status | string | false | none | エージェントのステータスは UP、DOWN、または UNKNOWN です |
| » alerts | object | false | none | エージェントに構成されたアラートのリスト。アラートステータスと確認されたアラートの数を含む |
| » total | integer | false | none | エージェントに構成されたアラートの数 |
| » acknowledged | integer | false | none | 確認されたアラートの数 |
| » high | integer | false | none | 状態が「高」のアラートの数 |

次のページに続く

表 204 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » medium | integer | false | none | 状態が「Medium」のアラートの数 |
| » low | integer | false | none | 状態が「Low」のアラートの数 |
| » high_acknowledged | integer | false | none | 状態が「高」の確認されたアラートの数 |
| » » medium_acknowledged | integer | false | none | 状態が「中」の確認されたアラートの数 |
| » low_acknowledged | integer | false | none | 状態が「Low」である確認されたアラートの数 |
| » version | string | false | none | エージェントの現在のバージョン |
| » processes | integer | false | none | 実行中のプロセスの数 |
| » threads | integer | false | none | 実行中のすべてのプロセスのスレッド数 |
| » cpu_utilization | number | false | none | CPU 使用率パーセント |
| » memory_utilization | number | false | none | メモリ使用率パーセント |
| » swap_utilization | number | false | none | スワップメモリ使用率パーセント |
| » disk_utilization | number | false | none | 全体のディスク使用率パーセント |

### 列挙値

| Property | Value |
|---|---|
| status | UP |
| status | DOWN |
| status | UNKNOWN |

**注釈**

この操作には認証が必要ありません

### get_*agent_status*{agent_id}

#### コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/agent/status/{agent_id} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/agent/status/{agent_id}, headers =␣
↪headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch("https://PEM-SERVER-IP/pem/api/v17/agent/status/{agent_id}", {
  method: "GET",

  headers: headers,
})
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /agent/status/{agent_id}

*agent_id* で識別される *pem* エージェントのステータスを取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| agent_id | path | integer | true | なし |
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例

```
{
  "id": 0,
  "name": "string",
  "group_id": 0,
  "group_name": "string",
```

(次のページに続く)

(前のページからの続き)


```
  "blackout": true,
  "status": "UP",
  "alerts": {
    "total": 0,
    "acknowledged": 0,
    "high": 0,
    "medium": 0,
    "low": 0,
    "high_acknowledged": 0,
    "medium_acknowledged": 0,
    "low_acknowledged": 0
  },
  "version": "string",
  "processes": 0,
  "threads": 0,
  "cpu_utilization": 0,
  "memory_utilization": 0,
  "swap_utilization": 0,
  "disk_utilization": 0
}
```


応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | エージェントステータス |
| 400 | Bad Request | エージェントが見つからないか、ユーザーに表示されない | None |

レスポンススキーマ

> **注釈**
>
> この操作には認証が必要ありません

### get_*server_status*

コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/server/status/ \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/server/status/, headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch("https://PEM-SERVER-IP/pem/api/v17/server/status/", {
  method: "GET",

  headers: headers,
})
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /server/status/

**表示されているすべてのデータベースサーバーのステータスを取得します。**

ユーザーに表示されるデータベースサーバーの情報と状態のリストを返しますトークンで表されます。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| X-Auth-Token | header | string | true | 認証用のトークン |

**応答例**

```
[
  {
    "id": 0,
    "name": "string",
    "group_id": 0,
    "group_name": "string",
    "blackout": true,
    "status": "UP",
    "alerts": {
      "total": 0,
      "acknowledged": 0,
      "high": 0,
      "medium": 0,
      "low": 0,
      "high_acknowledged": 0,
      "medium_acknowledged": 0,
      "low_acknowledged": 0
    },
    "version": "string",
    "number_connections": 0,
    "remote_monitoring": true,
    "agent_id": 0,
    "sessions": {
      "last_recorded_time": 0,
      "sessions": [
        {
          "database_name": "string",
          "procpid": 0,
          "usename": "string",
          "backend_start": 0,
          "xact_start": 0,
          "query_start": 0,
          "is_waiting": true,
          "is_idle": true,
          "is_idle_in_transaction": true,
          "is_vacuum": true,
          "is_autovacuum": true,
          "client_addr": "string",
          "client_port": 0,
          "memory_usage_mb": 0,
          "swap_usage_mb": 0,
          "cpu_usage": 0,
          "io_read_bytes": 0,
          "io_write_bytes": 0,
          "state": "string",
```


(次のページに続く)

(前のページからの続き)


```
          "state_change": 0
        }
      ]
    }
  }
]
```


### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | データベースサーバーの情報とステータスのリストを返します。 | Inline |

### レスポンススキーマ

ステータスコード **200**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| _anonymous_ | *Platform Compatibility* ] | false | none | なし |
| » id | integer | false | none | サーバー ID |
| » name | string | false | none | サーバー名 |
| » group_id | integer | false | none | データベースサーバーがユーザーに表示されるグループ ID |
| » group_name | string | false | none | グループ名 |
| » blackout | boolean | false | none | データベースサーバーのアラートブラックアウトのステータス |
| » status | string | false | none | サーバーのステータスは UP、DOWN、UNKNOWN、または UNMANAGED です。 |
| » alerts | object | false | none | データベースサーバーに構成されているアラートのリスト。アラートステータスと確認されたアラートの数を含む |
| » total | integer | false | none | データベースサーバーに構成されているアラートの数 |
| » acknowledged | integer | false | none | 確認されたアラートの数 |
| » high | integer | false | none | 状態が「高」のアラートの数 |

次のページに続く

表 210 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » medium | integer | false | none | 状態が「Medium」のアラートの数 |
| » low | integer | false | none | 状態が「Low」のアラートの数 |
| » high_acknowledged | integer | false | none | 状態が「高」の確認されたアラートの数 |
| » » medium_acknowledged | integer | false | none | 状態が「中」の確認されたアラートの数 |
| » low_acknowledged | integer | false | none | 状態が「Low」である確認されたアラートの数 |
| » version | string | false | none | データベースサーバーの現在のバージョン |
| » number_connections | integer | false | none | データベースサーバーの接続数 |
| » remote_monitoring | boolean | false | none | データベースサーバーが pemAgent によってリモートで監視されているかどうかを判断します |
| » agent_id | integer | false | none | データベースサーバーを監視しているエージェント ID |
| » sessions | object | false | none | データベースサーバーのすべてのセッション情報 |
| » last_recorded_time | number | false | none | セッション情報の最後に記録された時間の UNIX 時間 |
| » sessions | [object] | false | none | データベースサーバーのすべてのセッション情報のリスト |
| » database_name | string | false | none | データベース名 |
| » procpid | integer | false | none | プロセス ID |
| » username | string | false | none | ユーザー名 |
| » backend_start | number | false | none | セッション開始時刻の UNIX 時間 |
| » xact_start | number | false | none | トランザクション開始時間の UNIX 時間 |
| » query_start | number | false | none | クエリ開始時間の UNIX 時間クエリが実行されている場合 |
| » is_waiting | boolean | false | none | セッションは待機中ですか？ |
| » is_idle | boolean | false | none | セッションはアイドルですか? |
| » » » is_idle_in_transaction | boolean | false | none | セッションはトランザクションでアイドル状態ですか? |
| » is_vacuum | boolean | false | none | これは VACUUM セッションですか？ |
| » is_autovacuum | boolean | false | none | これは AUTO VACUUM セッションですか? |
| » client_addr | string | false | none | セッションのクライアントのホストアドレス |

次のページに続く

表 210 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » client_port | integer | false | none | セッションのクライアントポート |
| » memory_usage_mb | number | false | none | セッションのメモリ使用量 MB 単位ローカルに監視される場合にのみ使用可能 |
| » swap_usage_mb | number | false | none | セッションのスワップメモリ使用量 MB 単位ローカルに監視される場合にのみ使用可能 |
| » cpu_usage | number | false | none | セッションの CPU 使用率パーセンテージローカルに監視される場合にのみ使用可能 |
| » io_read_bytes | number | false | none | セッションの I/O 読み取りバイト単位ローカルに監視される場合にのみ使用可能 |
| » io_write_bytes | number | false | none | セッションの I/O 書き込みバイト単位ローカルに監視される場合にのみ使用可能 |
| » state | string | false | none | セッションの現在の状態アイドル、アクティブ、トランザクション中のアイドルなど |
| » state_change | number | false | none | セッション内の最後の状態変更の UNIX 時間 |

### 列挙値

| Property | Value |
|---|---|
| status | UP |
| status | DOWN |
| status | UNKNOWN |
| status | UNMANAGED |

**注釈**

この操作には認証が必要ありません

### get_*server_status*{server_id}

#### コードサンプル

```
#  You can also use wget
```

(次のページに続く)

(前のページからの続き)

```
curl -X GET https://PEM-SERVER-IP/pem/api/v17/server/status/{server_id} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/server/status/{server_id}, headers
↪= headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch("https://PEM-SERVER-IP/pem/api/v17/server/status/{server_id}", {
  method: "GET",

  headers: headers,
})
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /server/status/{server_id}

*server_id* で識別されるデータベースサーバーのステータスを取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | なし |
| X-Auth-Token | header | string | true | 承認トークン |

**応答例**

```
{
  "id": 0,
  "name": "string",
  "group_id": 0,
  "group_name": "string",
  "blackout": true,
  "status": "UP",
  "alerts": {
    "total": 0,
    "acknowledged": 0,
    "high": 0,
    "medium": 0,
    "low": 0,
    "high_acknowledged": 0,
    "medium_acknowledged": 0,
    "low_acknowledged": 0
  },
  "version": "string",
  "number_connections": 0,
  "remote_monitoring": true,
  "agent_id": 0,
  "sessions": {
    "last_recorded_time": 0,
    "sessions": [
      {
        "database_name": "string",
        "procpid": 0,
        "usename": "string",
        "backend_start": 0,
        "xact_start": 0,
        "query_start": 0,
        "is_waiting": true,
        "is_idle": true,
        "is_idle_in_transaction": true,
        "is_vacuum": true,
        "is_autovacuum": true,
        "client_addr": "string",
        "client_port": 0,
        "memory_usage_mb": 0,
        "swap_usage_mb": 0,
        "cpu_usage": 0,
        "io_read_bytes": 0,
        "io_write_bytes": 0,
        "state": "string",
        "state_change": 0
```


(次のページに続く)

(前のページからの続き)

```
      }
    ]
  }
}
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | *ServerStatus* |
| 400 | Bad Request | サーバーが見つからないか、ユーザーに表示されません | None |

### レスポンススキーマ

**注釈**

この操作には認証が必要ありません

## Agent Job

### get_*agent_job*{agent_id}

### コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/agent/job/{agent_id} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/agent/job/{agent_id}, headers =␣
↪headers)
```

(次のページに続く)

(前のページからの続き)

```
print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch("https://PEM-SERVER-IP/pem/api/v17/agent/job/{agent_id}", {
  method: "GET",

  headers: headers,
})
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /agent/job/{agent_id}

agent_id でエージェントレベルの Jobs を取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| agent_id | path | integer | true | エージェント ID |
| X-Auth-Token | header | string | true | 承認トークン |

#### 応答例

```
[
  {
    "jobname": "string",
    "jobenabled": true,
    "jobdesc": "string"
  }
]
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | Inline |

### レスポンススキーマ

ステータスコード **200**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| _anonymous_ | *Platform Compatibility* ] | false | none | なし |
| » jobname | string | false | none | ジョブの名前。 |
| » jobenabled | boolean | false | none | ジョブが有効か無効かを判断します。 |
| » jobdesc | string | false | none | ジョブの説明。 |

**注釈**

この操作には認証が必要ありません

### post_*agent_job*{agent_id}

#### コードサンプル

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v17/agent/job/{agent_id} \
  -H Content-Type: application/json \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Content-Type: application/json,
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.post(https://PEM-SERVER-IP/pem/api/v17/agent/job/{agent_id}, headers =␣
↪headers)

print(r.json())
```

```
const inputBody = {
  "jobname": "string",
  "jobenabled": true,
  "jobdesc": "string",
  "jsteps": [],
  "jschedules": [
    {
      "jscname": "string",
      "jscdesc": "string",
      "jscenabled": true,
      "jscstart": "2023-10-09 22:43 +05:30",
      "jscend": "2023-10-11 22:43 +05:30",
      "jscweekdays": [
        "Friday"
      ],
      "jscmonthdays": [
        1,
        15
      ],
      "jscmonths": [
        "January",
        "February",
        "March",
        "April",
        "May",
        "June",
        "July",
        "August",
        "September",
        "October",
        "November",
        "December"
      ],
      "jschours": [
        16
      ],
      "jscminutes": [
        0,
        30
      ],
      "jscexceptions": [
        {
          "jexdate": "2023-10-19",
          "jextime": "09:39:00"
        }
```

(次のページに続く)

(前のページからの続き)

```
      ],
      "jsctimezone": "America/New_York"
    }
  ],
  "notify": "DEFAULT",
  "email_group_id": "string"
};
const headers = {
  Content-Type:application/json,
  Accept:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v17/agent/job/{agent_id},
{
  method: POST,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

POST /agent/job/{agent_id}

エージェントレベルの Job を作成します。

**ボディパラメーター**

```
{
  "jobname": "string",
  "jobenabled": true,
  "jobdesc": "string",
  "jsteps": [],
  "jschedules": [
    {
      "jscname": "string",
      "jscdesc": "string",
      "jscenabled": true,
      "jscstart": "2023-10-09 22:43 +05:30",
      "jscend": "2023-10-11 22:43 +05:30",
      "jscweekdays": ["Friday"],
      "jscmonthdays": [1, 15],
      "jscmonths": [
```

(次のページに続く)

(前のページからの続き)


```
        "January",
        "February",
        "March",
        "April",
        "May",
        "June",
        "July",
        "August",
        "September",
        "October",
        "November",
        "December"
      ],
      "jschours": [16],
      "jscminutes": [0, 30],
      "jscexceptions": [
        {
          "jexdate": "2023-10-19",
          "jextime": "09:39:00"
        }
      ],
      "jsctimezone": "America/New_York"
    }
  ],
  "notify": "DEFAULT",
  "email_group_id": "string"
}
```


### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| agent_id | path | integer | true | エージェント ID |
| X-Auth-Token | header | string | true | 承認トークン |
| body | body | *AgentJobBody-Post* | true | なし |

### 応答例


```
{
  "id": 0
}
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 201 | *8.* スケジュールされた停電が作成されましたダイアログ | 正常に作成されました | Inline |

### レスポンススキーマ

ステータスコード **201**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » id | integer | false | none | ジョブの ID |

**注釈**

この操作には認証が必要ありません

### get_*agent_job*{agent_id}_{job_id}

### コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/agent/job/{agent_id}/{job_id} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/agent/job/{agent_id}/{job_id},␣
↪headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
```

(次のページに続く)

(前のページからの続き)

```
};

fetch("https://PEM-SERVER-IP/pem/api/v17/agent/job/{agent_id}/{job_id}", {
  method: "GET",

  headers: headers,
})
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /agent/job/{agent_id}/{job_id}

agent_id および job_id でエージェントレベルの Jobs を取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| agent_id | path | integer | true | エージェント ID |
| job_id | path | integer | true | ジョブ ID |
| X-Auth-Token | header | string | true | 承認トークン |

#### 応答例

```
{
  "jobid": 0,
  "jobname": "string",
  "jobenabled": true,
  "jobdesc": "string",
  "jobcreated": "2019-08-24T14:15:22Z",
  "jobchanged": "2019-08-24T14:15:22Z",
  "jlgstatus": "string",
  "jobnextrun": "2019-08-24T14:15:22Z",
  "joblastrun": "2019-08-24T14:15:22Z",
  "is_alert_job": true,
  "notify": "string",
  "email_group_id": "string",
  "jsteps": [],
  "jschedules": [
    {
      "jscid": 0,
```

(次のページに続く)

(前のページからの続き)


```
    "jscjobid": 0,
    "jscname": "string",
    "jscdesc": "string",
    "jscenabled": true,
    "jscstart": "2019-08-24T14:15:22Z",
    "jscend": "2019-08-24T14:15:22Z",
    "jscweekdays": [false, false, false, false, false, true, false],
    "jscmonthdays": [
      false,
      false,
      false,
      false,
      false,
      false,
      false,
      false,
      true,
      false,
      false,
      false,
      false,
      false,
      false,
      false,
      false,
      false,
      false,
      false,
      false,
      false,
      false,
      false,
      false,
      false,
      false,
      false,
      false,
      false,
      false,
      false,
      false
    ],
    "jscmonths": [
      true,
      false,
      false,
```


(次のページに続く)

(前のページからの続き)

```
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false
      ],
      "jschours": [
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        true,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false
      ],
      "jscminutes": [
        false,
        false,
        false,
        false,
        true,
        false,
        false,
        false,
```

(次のページに続く)

(前のページからの続き)

```
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
```

(次のページに続く)

(前のページからの続き)

```
        false,
        false,
        false,
        false,
        false,
        false,
        false
      ],
      "jscexceptions": [null],
      "jsctimezone": "string"
    }
  ]
}
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | *AgentJobBodyGet* |

**注釈**

この操作には認証が必要ありません

### put_*agent_job*{agent_id}_{job_id}

#### コードサンプル

```
#  You can also use wget

curl -X PUT https://PEM-SERVER-IP/pem/api/v17/agent/job/{agent_id}/{job_id} \
  -H Content-Type: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Content-Type: application/json,
  X-Auth-Token: string
}
```

(次のページに続く)

(前のページからの続き)

```
r = requests.put(https://PEM-SERVER-IP/pem/api/v17/agent/job/{agent_id}/{job_id},␣
↪headers = headers)

print(r.json())
```

```
const inputBody = {
  "jobname": "string",
  "jobenabled": true,
  "jobdesc": "string",
  "jsteps": {
    "added": [
      {
        "jstname": "string",
        "jstdesc": "string",
        "jstenabled": true,
        "jstkind": true,
        "jstcode": "string",
        "server_id": 0,
        "database_name": "string",
        "jstonerror": "f",
        "report_template_id": 0
      }
    ],
    "changed": [
      {
        "jstid": 0,
        "jstname": "string",
        "jstdesc": "string",
        "jstenabled": true,
        "jstkind": true,
        "jstcode": "string",
        "server_id": 0,
        "database_name": "string",
        "jstonerror": "f",
        "report_template_id": 0,
        "jstnextrun": "string"
      }
    ]
  },
  "jschedules": {
    "added": [
      {
        "jscname": "string",
        "jscdesc": "string",
```

(次のページに続く)

(前のページからの続き)


```
        "jscenabled": true,
        "jscstart": "2023-10-09 22:43 +05:30",
        "jscend": "2023-10-11 22:43 +05:30",
        "jscweekdays": [
          "Friday"
        ],
        "jscmonthdays": [
          1,
          15
        ],
        "jscmonths": [
          "January",
          "February",
          "March",
          "April",
          "May",
          "June",
          "July",
          "August",
          "September",
          "October",
          "November",
          "December"
        ],
        "jschours": [
          16
        ],
        "jscminutes": [
          0,
          30
        ],
        "jscexceptions": [
          {
            "jexdate": "2023-10-19",
            "jextime": "09:39:00"
          }
        ],
        "jsctimezone": "America/New_York"
      }
    ],
    "changed": [
      {
        "jscid": 0,
        "jscname": "string",
        "jscdesc": "string",
```


(次のページに続く)

(前のページからの続き)

```
        "jscenabled": true,
        "jscstart": "2023-10-09 22:43 +05:30",
        "jscend": "2023-10-11 22:43 +05:30",
        "jscweekdays": [
          "Friday"
        ],
        "jscmonthdays": [
          1,
          15
        ],
        "jscmonths": [
          "January",
          "February",
          "March",
          "April",
          "May",
          "June",
          "July",
          "August",
          "September",
          "October",
          "November",
          "December"
        ],
        "jschours": [
          16
        ],
        "jscminutes": [
          0,
          30
        ],
        "jscexceptions": [
          {
            "jexdate": "2023-10-19",
            "jextime": "09:39:00"
          }
        ],
        "jsctimezone": "America/New_York"
      }
    ]
  },
  "notify": "DEFAULT",
  "email_group_id": "string"
};
const headers = {
```

(次のページに続く)

(前のページからの続き)

```
  Content-Type:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v17/agent/job/{agent_id}/{job_id},
{
  method: PUT,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

PUT /agent/job/{agent_id}/{job_id}

エージェントレベルの Job を更新します。

**ボディパラメーター**

```
{
  "jobname": "string",
  "jobenabled": true,
  "jobdesc": "string",
  "jsteps": {
    "added": [
      {
        "jstname": "string",
        "jstdesc": "string",
        "jstenabled": true,
        "jstkind": true,
        "jstcode": "string",
        "server_id": 0,
        "database_name": "string",
        "jstonerror": "f",
        "report_template_id": 0
      }
    ],
    "changed": [
      {
        "jstid": 0,
        "jstname": "string",
        "jstdesc": "string",
        "jstenabled": true,
```

(次のページに続く)

(前のページからの続き)


```
      "jstkind": true,
      "jstcode": "string",
      "server_id": 0,
      "database_name": "string",
      "jstonerror": "f",
      "report_template_id": 0,
      "jstnextrun": "string"
    }
  ]
},
"jschedules": {
  "added": [
    {
      "jscname": "string",
      "jscdesc": "string",
      "jscenabled": true,
      "jscstart": "2023-10-09 22:43 +05:30",
      "jscend": "2023-10-11 22:43 +05:30",
      "jscweekdays": ["Friday"],
      "jscmonthdays": [1, 15],
      "jscmonths": [
        "January",
        "February",
        "March",
        "April",
        "May",
        "June",
        "July",
        "August",
        "September",
        "October",
        "November",
        "December"
      ],
      "jschours": [16],
      "jscminutes": [0, 30],
      "jscexceptions": [
        {
          "jexdate": "2023-10-19",
          "jextime": "09:39:00"
        }
      ],
      "jsctimezone": "America/New_York"
    }
  ],
```


(次のページに続く)

(前のページからの続き)


```
    "changed": [
      {
        "jscid": 0,
        "jscname": "string",
        "jscdesc": "string",
        "jscenabled": true,
        "jscstart": "2023-10-09 22:43 +05:30",
        "jscend": "2023-10-11 22:43 +05:30",
        "jscweekdays": ["Friday"],
        "jscmonthdays": [1, 15],
        "jscmonths": [
          "January",
          "February",
          "March",
          "April",
          "May",
          "June",
          "July",
          "August",
          "September",
          "October",
          "November",
          "December"
        ],
        "jschours": [16],
        "jscminutes": [0, 30],
        "jscexceptions": [
          {
            "jexdate": "2023-10-19",
            "jextime": "09:39:00"
          }
        ],
        "jsctimezone": "America/New_York"
      }
    ]
  },
  "notify": "DEFAULT",
  "email_group_id": "string"
}
```

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| agent_id | path | integer | true | エージェント ID |
| job_id | path | integer | true | ジョブ ID |
| X-Auth-Token | header | string | true | 承認トークン |
| body | body | *AgentJobBody-Put* | true | なし |

### 応答例

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ステータス 200 | None |

### レスポンススキーマ

**注釈**

この操作には認証が必要ありません

## delete_*agent_job*{agent_id}_{job_id}

### コードサンプル

```
#  You can also use wget

curl -X DELETE https://PEM-SERVER-IP/pem/api/v17/agent/job/{agent_id}/{job_id} \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  X-Auth-Token: string
}

r = requests.delete(https://PEM-SERVER-IP/pem/api/v17/agent/job/{agent_id}/{job_id},␣
↪headers = headers)

print(r.json())
```

```
const headers = {
  "X-Auth-Token": "string",
};

fetch("https://PEM-SERVER-IP/pem/api/v17/agent/job/{agent_id}/{job_id}", {
  method: "DELETE",

  headers: headers,
})
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

`DELETE /agent/job/{agent_id}/{job_id}`

agent_id と job_id を使用して Job を削除します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| agent_id | path | integer | true | エージェント ID |
| job_id | path | integer | true | ジョブ ID |
| X-Auth-Token | header | string | true | 承認トークン |

**応答例**

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 削除が成功しました。 | None |

### レスポンススキーマ

**注釈**

この操作には認証が必要ありません

### Alerts History

#### get_alert_history

コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/alert/history/ \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/alert/history/, headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch("https://PEM-SERVER-IP/pem/api/v17/alert/history/", {
  method: "GET",

  headers: headers,
})
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /alert/history/

すべてのアラートの状態変更履歴を取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例

```
[
  {
    "alert_id": 0,
    "state": "High",
    "value": "string",
    "actual_value": 0
  }
]
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | Inline |

### レスポンススキーマ

ステータスコード **200**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| _anonymous_ | *Platform Compatibility* ] | false | none | なし |
| » alert_id | integer | false | none | アラート ID |
| » state | string | false | none | 州 |
| » value | string | false | none | ユーザーに提示できる価値 |
| » actual_value | number | false | none | 状態変更時の実際の値 |

### 列挙値

| Property | Value |
|---|---|
| state | High |
| state | Medium |
| state | Low |

**注釈**

この操作には認証が必要ありません

### get_*alert_history_agent*{agent_id}

#### コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/alert/history/agent/{agent_id} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/alert/history/agent/{agent_id},␣
↪headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch("https://PEM-SERVER-IP/pem/api/v17/alert/history/agent/{agent_id}", {
  method: "GET",

  headers: headers,
})
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
```

(次のページに続く)

(前のページからの続き)

```
  console.log(body);
});
```

GET /alert/history/agent/{agent_id}

エージェントのすべてのアラートの状態変更履歴を取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| agent_id | path | integer | true | 有効なエージェント ID |
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例

```
[
  {
    "alert_id": 0,
    "state": "High",
    "value": "string",
    "actual_value": 0
  }
]
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | Inline |

### レスポンススキーマ

ステータスコード **200**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| _anonymous_ | *Platform Compatibility* ] | false | none | なし |
| » alert_id | integer | false | none | アラート ID |
| » state | string | false | none | 州 |

次のページに続く

表 232 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » value | string | false | none | ユーザーに提示できる価値 |
| » actual_value | number | false | none | 状態変更時の実際の値 |

### 列挙値

| Property | Value |
|---|---|
| state | High |
| state | Medium |
| state | Low |

**注釈**

この操作には認証が必要ありません

### get_*alert_history_server*{server_id}

#### コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/alert/history/server/{server_id} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/alert/history/server/{server_id},␣
↪headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};
```

(次のページに続く)

(前のページからの続き)

```
fetch("https://PEM-SERVER-IP/pem/api/v17/alert/history/server/{server_id}", {
  method: "GET",

  headers: headers,
})
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /alert/history/server/{server_id}

データベースサーバーのすべてのアラートの状態変更履歴を取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | 有効なサーバー ID |
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例

```
[
  {
    "alert_id": 0,
    "state": "High",
    "value": "string",
    "actual_value": 0
  }
]
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | Inline |

### レスポンススキーマ

ステータスコード **200**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| _anonymous_ | *Platform Compatibility* ] | false | none | なし |
| » alert_id | integer | false | none | アラート ID |
| » state | string | false | none | 州 |
| » value | string | false | none | ユーザーに提示できる価値 |
| » actual_value | number | false | none | 状態変更時の実際の値 |

### 列挙値

| Property | Value |
|---|---|
| state | High |
| state | Medium |
| state | Low |

**注釈**

この操作には認証が必要ありません

## get_*alert_history_server*{server_id}*database*{database_name}

### コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/alert/history/server/{server_id}/
↪database/{database_name} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}
```

(次のページに続く)

(前のページからの続き)

```
r = requests.get(https://PEM-SERVER-IP/pem/api/v17/alert/history/server/{server_id}/
↪database/{database_name}, headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch(
  "https://PEM-SERVER-IP/pem/api/v17/alert/history/server/{server_id}/database/
↪{database_name}",
  {
    method: "GET",

    headers: headers,
  },
)
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /alert/history/server/{server_id}/database/{database_name}

データベースサーバーのデータベースのすべてのアラートの状態変更履歴を取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | 有効なサーバー ID |
| database_name | path | string | true | データベース名 |
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例

```
[
  {
    "alert_id": 0,
    "state": "High",
    "value": "string",
```

(次のページに続く)

(前のページからの続き)

```
    "actual_value": 0
  }
]
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | Inline |

### レスポンススキーマ

ステータスコード **200**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| _anonymous_ | *Platform Compatibility* ] | false | none | なし |
| » alert_id | integer | false | none | アラート ID |
| » state | string | false | none | 州 |
| » value | string | false | none | ユーザーに提示できる価値 |
| » actual_value | number | false | none | 状態変更時の実際の値 |

### 列挙値

| Property | Value |
|---|---|
| state | High |
| state | Medium |
| state | Low |

**注釈**

この操作には認証が必要ありません

### Alerts Status

#### get_*alert_status*

コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/alert/status/ \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/alert/status/, headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch("https://PEM-SERVER-IP/pem/api/v17/alert/status/", {
  method: "GET",

  headers: headers,
})
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /alert/status/

すべてのアラートの現在のステータスを取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例


```
[
  {
    "alert_since": "string",
    "alert_id": 0,
    "alert_name": "string",
    "alert_state": "High",
    "value": "string",
    "object_description": "string",
    "server_id": 0,
    "agent_id": 0,
    "database": "string",
    "schema": "string",
    "package": "string",
    "object": "string",
    "alert_target_level": "Global",
    "last_processed": "string",
    "info": "string",
    "info_cols": ["string"],
    "info_vals": [["string"]]
  }
]
```


### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | Inline |

### レスポンススキーマ

ステータスコード **200**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| _anonymous_ | *Platform Compatibility* ] | false | none | なし |
| » alert_since | string | false | none | アラート状態が最後に変更された Unix 時間 |
| » alert_id | integer | false | none | アラート内部識別番号 |
| » alert_name | string | false | none | アラートの名前 |
| » alert_state | string | false | none | アラートの現在の状態。'null' 値は、アラートのアラートしきい値が現在違反していないことを示します。 |
| » value | string | false | none | アラートの現在値 |
| » object_description | string | false | none | モニタリングオブジェクトの説明/名前 |
| » server_id | integer | false | none | null またはサーバーの ID |
| » agent_id | integer | false | none | null またはエージェントの ID |
| » database | string | false | none | null またはデータベースの名前 |
| » schema | string | false | none | null またはスキーマ/名前空間の名前 |
| » package | string | false | none | null または EDB パッケージの名前 |
| » object | string | false | none | null または他の種類のオブジェクトの名前 |
| » alert_target_level | string | false | none | 警戒対象レベル |
| » last_processed | string | false | none | アラートが処理された Unix 時間を表す文字列 |
| » info | string | false | none | アラートが生成された時点の情報 |
| » info_cols | [string] | false | none | 情報の列名の配列 |
| » info_vals | [array] | false | none | テーブル形式のアラートに関連付けられた詳細情報配列の配列 |

### 列挙値

| Property | Value |
|---|---|
| alert_state | High |
| alert_state | Medium |
| alert_state | Low |

次のページに続く

表 245 – 前のページからの続き

| Property | Value |
|---|---|
| alert_target_level | Global |
| alert_target_level | Agent |
| alert_target_level | Server |
| alert_target_level | Database |
| alert_target_level | Schema |
| alert_target_level | Table |
| alert_target_level | Index |
| alert_target_level | Sequence |
| alert_target_level | Function |
| alert_target_level | View |

**注釈**

この操作には認証が必要ありません

### get_*alert_status*{since}

#### コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/alert/status/{since} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/alert/status/{since}, headers =␣
↪headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch("https://PEM-SERVER-IP/pem/api/v17/alert/status/{since}", {
```

(次のページに続く)

(前のページからの続き)

```
  method: "GET",

  headers: headers,
})
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /alert/status/{since}

指定された時間後のアラート状態の変更のリストを取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| since | path | number | true | Unix タイムライン |
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例

```
{
  "alert_since": "string",
  "alert_id": 0,
  "alert_name": "string",
  "alert_state": "High",
  "value": "string",
  "object_description": "string",
  "server_id": 0,
  "agent_id": 0,
  "database": "string",
  "schema": "string",
  "package": "string",
  "object": "string",
  "alert_target_level": "Global",
  "last_processed": "string",
  "info": "string",
  "info_cols": ["string"],
  "info_vals": [["string"]]
}
```

応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | アラートステータス |

**注釈**

この操作には認証が必要ありません

### Alerts

#### get_alert_config_global

コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/alert/config/global/ \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/alert/config/global/, headers =␣
↪headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch("https://PEM-SERVER-IP/pem/api/v17/alert/config/global/", {
  method: "GET",

  headers: headers,
```

(次のページに続く)

(前のページからの続き)

```
})
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

```
GET /alert/config/global/
```

グローバルレベルの Alerts を取得します。 name 完全一致または name_contains 部分文字列検索クエリパラメーターを介してオプションの名前ベースのフィルタリングをサポートします。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| X-Auth-Token | header | string | true | 承認トークン |
| name | query | string | false | 名前の完全な一致大文字と小文字を区別してアラートをフィルタリングします。 name_contains と相互排他的。 |
| name_contains | query | string | false | 部分文字列の一致大文字と小文字を区別してアラートをフィルタリングします。 name と相互排他的。 |

### 応答例

```
[
  {
    "id": 0,
    "alert_name": "string",
    "alert_template": 0,
    "description": "string",
    "enabled": true,
    "history_retention": 0,
    "frequency_min": 0,
    "auto_created": true,
    "template_id": "string",
    "params": [],
    "param_names": ["string"],
    "param_types": "string",
    "params_units": ["string"],
    "threshold_unit": "string",
    "operator": "string",
```

(次のページに続く)

(前のページからの続き)


```
    "low_threshold_value": 0,
    "medium_threshold_value": 0,
    "high_threshold_value": 0,
    "send_email": true,
    "all_alert_enable": true,
    "email_group_id": 0,
    "low_alert_enable": true,
    "low_email_group_id": 0,
    "med_alert_enable": true,
    "med_email_group_id": 0,
    "high_alert_enable": true,
    "high_email_group_id": 0,
    "send_trap": true,
    "snmp_trap_version": 0,
    "low_send_trap": true,
    "med_send_trap": true,
    "high_send_trap": true,
    "cleared_alert_enable": true,
    "execute_script": true,
    "execute_script_on_clear": true,
    "execute_script_on_pem_server": true,
    "script_code": "string",
    "submit_to_nagios": true,
    "override_default_config": true,
    "send_notification": true,
    "low_webhook_ids": [0],
    "med_webhook_ids": [0],
    "high_webhook_ids": [0],
    "cleared_webhook_ids": [0]
  }
]
```


応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | Inline |

次のページに続く

表 249 – 前のページからの続き

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 400 | Bad Request | 相互排他性違反 `name`または類似の完全一致フィルタと`name_contains`または類似の部分文字列フィルタが両方とも同じ要求で指定されました。フィルタリングは、ペアのうちの 1 つのみを受け入れます。 | Inline |

### レスポンススキーマ

ステータスコード **200**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| _anonymous_ | *Platform Compatibility* ] | false | none | なし |
| » id | integer | false | none | アラート ID。注このプロパティは、POST 要求には適用されません。 |
| » alert_name | string | false | none | アラートの名前。注 このプロパティは、POST 要求の必須パラメーターです。 |
| » alert_template | integer | false | none | アラートテンプレートの ID。このプロパティは、正の整数値である必要があります。注これはPOST 要求の必須パラメーターです。 |
| » description | string | false | none | なし |
| » enabled | boolean | false | none | このアラートが有効/無効かどうかを判断します。注 このプロパティは、POST 要求の必須パラメーターです。 |

次のページに続く

表 250 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » history_retention | integer | false | none | このアラートの履歴を保持する日数を指定します。指定できる値は 1 ～99999 です。 注このプロパティは、POST 要求の必須パラメーターです。 |
| » frequency_min | integer | false | none | PEM データベースサーバーでアラートを確認する頻度を分単位で定義します。指定できる値は 1 ～65534 です。 注このプロパティは、POST 要求の必須パラメーターです。 |
| » auto_created | boolean | false | none | アラートを自動作成するかどうかを指定します。 |
| » template_id | string | false | none | アラートテンプレートを定義します。 |
| » params | *Platform Compatibility* ] | false | none | アラートを確認するためにアラートテンプレートが使用するパラメーター/値のペアのリスト。注アラートテンプレートには、all パラメーターを定義する必要があります。 |
| » paramname | string | false | none | パラメーターの名前。注このプロパティは、アラートテンプレートパラメーターリストに存在する必要があります。例 |
| » paramvalue | string | false | none | 指定されたパラメーターの値 |
| » param_names | [string] | false | none | パラメーターの名前を定義します。 |
| » param_types | string | false | none | パラメーターのタイプを定義します。例 INTEGER/STRING.. |
| » params_units | [string] | false | none | パラメーターに使用される単位を定義します。例 'DAYS'、'True or False'、'Hours'、'Minutes' など |

次のページに続く

表 250 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » threshold_unit | string | false | none | MB/kB/秒/日/... などの、しきい値の計算に使用される単位 この値は、GUI でのみ使用されます。 |
| » operator | string | false | none | しきい値をチェックするための演算子を定義します。有効な文字列は>"および"<"です。注 このプロパティは、POST 要求の必須パラメーターです。 |
| » low_threshold_value | number | false | none | 低アラートをトリガーするためのしきい値。このプロパティには、有効な整数/浮動小数点値を指定できます。注これは POST 要求の必須パラメーターです。 |
| » medium_threshold_value | number | false | none | 中アラートをトリガーするためのしきい値。このプロパティには、有効な整数/浮動小数点値を指定できます。注これは POST 要求の必須パラメーターです。 |
| » high_threshold_value | number | false | none | 高度なアラートをトリガーするためのしきい値。任意の有効な整数/浮動小数点値を指定できます。注これは POST 要求の必須パラメーターです。 |
| » send_email | boolean | false | none | アラートの状態が変更されたときに電子メールを送信するかどうかを決定します。 |
| » all_alert_enable | boolean | false | none | アラートの状態が変更されたときに、パラメーター 'email_group_id' で指定されたグループにメール通知を送信するかどうかを決定します。 |

次のページに続く

表 250 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » email_group_id | integer | false | none | アラートの状態が変更されたときに電子メール通知の送信先の電子メールグループを定義します。 |
| » low_alert_enable | boolean | false | none | アラートの状態が LOW に設定されているときに、パラメーター 'low_email_group_id' で指定されたグループにメール通知を送信するかどうかを決定します。 |
| » low_email_group_id | integer | false | none | アラートの状態が LOW に変更されたときに電子メール通知の送信先の電子メールグループを定義します。 |
| » med_alert_enable | boolean | false | none | アラートの状態が MEDIUM に設定されている場合、パラメーター 'medium_email_group_id' で指定されたグループにメール通知を送信するかどうかを決定します。 |
| » med_email_group_id | integer | false | none | アラートの状態が MEDIUM に変更されたときにメール通知の送信先のメールグループを定義します。 |
| » high_alert_enable | boolean | false | none | アラートの状態が HIGH に設定されているときに、パラメーター 'high_email_group_id' で指定されたグループに電子メール通知を送信するかどうかを決定します。 |
| » high_email_group_id | integer | false | none | アラートの状態が HIGH に変更されたときに電子メール通知の送信先の電子メールグループを定義します。 |

次のページに続く

表 250 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » send_trap | boolean | false | none | アラートの状態が変更されたときに SNMP トラップ通知を送信するかどうかを決定します。 |
| » snmp_trap_version | integer | false | none | SNMP トラップバージョンを定義します。サポートされている SNMP トラップバージョンは 1、2、または 3 です |
| » low_send_trap | boolean | false | none | アラートの状態が LOW に変更されたときに SNMP トラップ通知を送信するかどうかを決定します。 |
| » med_send_trap | boolean | false | none | アラートの状態が MEDIUM に変更されたときに SNMP トラップ通知を送信するかどうかを決定します。 |
| » high_send_trap | boolean | false | none | アラートの状態が HIGH に変わったときに SNMP トラップ通知を送信するかどうかを決定します。 |
| » cleared_alert_enable | boolean | false | none | 'クリアアラート' の通知をミュートするかどうかを決定します。 |
| » execute_script | boolean | false | none | アラートの状態が LOW/MEDIUM/HIGH に変更された場合に、提供されたスクリプトを実行するかどうかを決定します。 |
| » execute_script_on_clear | boolean | false | none | アラートの状態がクリアされた場合に、提供されたスクリプトを実行するかどうかを決定します。 |

次のページに続く

表 250 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » execute_script_on_pem_serve | boolean | false | none | 提供されたスクリプトをPEM ホストまたはエージェントがオブジェクトを監視しているホストで実行するかどうかを決定します。 |
| » script_code | string | false | none | 実行するシェルまたはバッチスクリプト。詳細については、ドキュメントをご覧ください。 |
| » submit_to_nagios | boolean | false | none | アラートの状態が変更されたときに nagios に通知を送信するかどうかを決定します。 |
| » override_default_config | boolean | false | none | デフォルトの Webhook 構成をオーバーライドするかどうかを決定します。 |
| » send_notification | boolean | false | none | Webhook 送信通知を有効にするかどうかを決定します。 |
| » low_webhook_ids | [integer] | false | none | 低アラートが構成されている Webhook ID |
| » med_webhook_ids | [integer] | false | none | 中アラートが構成されている Webhook ID |
| » high_webhook_ids | [integer] | false | none | 高アラートが構成されている Webhook ID |
| » cleared_webhook_ids | [integer] | false | none | クリアされたアラートが構成されている Webhook ID |

### 列挙値

| Property | Value |
|---|---|
| paramname | Exclude tables smaller than |
| paramname | mount point |
| paramname | Host IP Address |
| paramname | Time interval to monitor |
| paramname | Time since idle |
| paramname | Per CPU utilization (K%) |

ステータスコード **400**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » success | integer | false | none | なし |
| » errormsg | string | false | none | なし |

**注釈**

この操作には認証が必要ありません

### post_alert_config_global

#### コードサンプル

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v17/alert/config/global/ \
  -H Content-Type: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Content-Type: application/json,
  X-Auth-Token: string
}

r = requests.post(https://PEM-SERVER-IP/pem/api/v17/alert/config/global/, headers =␣
↪headers)

print(r.json())
```

```
const inputBody = {
  "alert_name": "string",
  "alert_template": 0,
  "description": "string",
  "enabled": true,
  "history_retention": 0,
  "frequency_min": 0,
  "operator": "string",
  "low_threshold_value": 0,
  "medium_threshold_value": 0,
  "high_threshold_value": 0,
  "all_alert_enable": true,
  "email_group_id": 1,
  "low_alert_enable": true,
```

(次のページに続く)

(前のページからの続き)

```
  "low_email_group_id": 1,
  "med_alert_enable": true,
  "med_email_group_id": 1,
  "high_alert_enable": true,
  "high_email_group_id": 1,
  "send_trap": false,
  "snmp_trap_version": 0,
  "low_send_trap": true,
  "med_send_trap": true,
  "high_send_trap": true,
  "cleared_alert_enable": true,
  "params": [],
  "execute_script": true,
  "execute_script_on_clear": true,
  "execute_script_on_pem_server": true,
  "script_code": "string",
  "submit_to_nagios": true,
  "override_default_config": true,
  "send_notification": false,
  "low_webhook_ids": [],
  "med_webhook_ids": [],
  "high_webhook_ids": [],
  "cleared_webhook_ids": []
};
const headers = {
  Content-Type:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v17/alert/config/global/,
{
  method: POST,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

POST /alert/config/global/

グローバルレベルの Alert を作成します。

**ボディパラメーター**

```
{
  "alert_name": "string",
  "alert_template": 0,
  "description": "string",
  "enabled": true,
  "history_retention": 0,
  "frequency_min": 0,
  "operator": "string",
  "low_threshold_value": 0,
  "medium_threshold_value": 0,
  "high_threshold_value": 0,
  "all_alert_enable": true,
  "email_group_id": 1,
  "low_alert_enable": true,
  "low_email_group_id": 1,
  "med_alert_enable": true,
  "med_email_group_id": 1,
  "high_alert_enable": true,
  "high_email_group_id": 1,
  "send_trap": false,
  "snmp_trap_version": 0,
  "low_send_trap": true,
  "med_send_trap": true,
  "high_send_trap": true,
  "cleared_alert_enable": true,
  "params": [],
  "execute_script": true,
  "execute_script_on_clear": true,
  "execute_script_on_pem_server": true,
  "script_code": "string",
  "submit_to_nagios": true,
  "override_default_config": true,
  "send_notification": false,
  "low_webhook_ids": [],
  "med_webhook_ids": [],
  "high_webhook_ids": [],
  "cleared_webhook_ids": []
}
```


### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| X-Auth-Token | header | string | true | 承認トークン |

次のページに続く

表 253 – 前のページからの続き

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| body | body | *AlertPayload* | true | なし |

#### 応答例

#### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ステータス 200 | None |

#### レスポンススキーマ

**注釈**

この操作には認証が必要ありません

### get_*alert_config_global*{alert_id}

#### コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/alert/config/global/{alert_id} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/alert/config/global/{alert_id},␣
↪headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch("https://PEM-SERVER-IP/pem/api/v17/alert/config/global/{alert_id}", {
  method: "GET",

  headers: headers,
})
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /alert/config/global/{alert_id}

agent_id でグローバルレベルの Alerts を取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| alert_id | path | integer | true | アラート ID |
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例

```
{
  "id": 0,
  "alert_name": "string",
  "alert_template": 0,
  "description": "string",
  "enabled": true,
  "history_retention": 0,
  "frequency_min": 0,
  "auto_created": true,
  "template_id": "string",
  "params": [],
  "param_names": ["string"],
  "param_types": "string",
  "params_units": ["string"],
  "threshold_unit": "string",
  "operator": "string",
```

(次のページに続く)

(前のページからの続き)


```
  "low_threshold_value": 0,
  "medium_threshold_value": 0,
  "high_threshold_value": 0,
  "send_email": true,
  "all_alert_enable": true,
  "email_group_id": 0,
  "low_alert_enable": true,
  "low_email_group_id": 0,
  "med_alert_enable": true,
  "med_email_group_id": 0,
  "high_alert_enable": true,
  "high_email_group_id": 0,
  "send_trap": true,
  "snmp_trap_version": 0,
  "low_send_trap": true,
  "med_send_trap": true,
  "high_send_trap": true,
  "cleared_alert_enable": true,
  "execute_script": true,
  "execute_script_on_clear": true,
  "execute_script_on_pem_server": true,
  "script_code": "string",
  "submit_to_nagios": true,
  "override_default_config": true,
  "send_notification": true,
  "low_webhook_ids": [0],
  "med_webhook_ids": [0],
  "high_webhook_ids": [0],
  "cleared_webhook_ids": [0]
}
```


応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | アラートスレッドについて |

**注釈**

この操作には認証が必要ありません

### put_*alert_config_global*{alert_id}

コードサンプル

```
#  You can also use wget

curl -X PUT https://PEM-SERVER-IP/pem/api/v17/alert/config/global/{alert_id} \
  -H Content-Type: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Content-Type: application/json,
  X-Auth-Token: string
}

r = requests.put(https://PEM-SERVER-IP/pem/api/v17/alert/config/global/{alert_id},␣
↪headers = headers)

print(r.json())
```

```
const inputBody = {
  "alert_name": "string",
  "alert_template": 0,
  "description": "string",
  "enabled": true,
  "history_retention": 0,
  "frequency_min": 0,
  "operator": "string",
  "low_threshold_value": 0,
  "medium_threshold_value": 0,
  "high_threshold_value": 0,
  "all_alert_enable": true,
  "email_group_id": 1,
  "low_alert_enable": true,
  "low_email_group_id": 1,
  "med_alert_enable": true,
  "med_email_group_id": 1,
  "high_alert_enable": true,
  "high_email_group_id": 1,
  "send_trap": false,
  "snmp_trap_version": 0,
  "low_send_trap": true,
  "med_send_trap": true,
  "high_send_trap": true,
  "cleared_alert_enable": true,
```

(次のページに続く)

(前のページからの続き)

```
  "params": [],
  "execute_script": true,
  "execute_script_on_clear": true,
  "execute_script_on_pem_server": true,
  "script_code": "string",
  "submit_to_nagios": true,
  "override_default_config": true,
  "send_notification": false,
  "low_webhook_ids": [],
  "med_webhook_ids": [],
  "high_webhook_ids": [],
  "cleared_webhook_ids": []
};
const headers = {
  Content-Type:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v17/alert/config/global/{alert_id},
{
  method: PUT,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

PUT /alert/config/global/{alert_id}

グローバルレベルの Alert を更新します。

**ボディパラメーター**

```
{
  "alert_name": "string",
  "alert_template": 0,
  "description": "string",
  "enabled": true,
  "history_retention": 0,
  "frequency_min": 0,
  "operator": "string",
  "low_threshold_value": 0,
  "medium_threshold_value": 0,
```

(次のページに続く)

(前のページからの続き)


```
  "high_threshold_value": 0,
  "all_alert_enable": true,
  "email_group_id": 1,
  "low_alert_enable": true,
  "low_email_group_id": 1,
  "med_alert_enable": true,
  "med_email_group_id": 1,
  "high_alert_enable": true,
  "high_email_group_id": 1,
  "send_trap": false,
  "snmp_trap_version": 0,
  "low_send_trap": true,
  "med_send_trap": true,
  "high_send_trap": true,
  "cleared_alert_enable": true,
  "params": [],
  "execute_script": true,
  "execute_script_on_clear": true,
  "execute_script_on_pem_server": true,
  "script_code": "string",
  "submit_to_nagios": true,
  "override_default_config": true,
  "send_notification": false,
  "low_webhook_ids": [],
  "med_webhook_ids": [],
  "high_webhook_ids": [],
  "cleared_webhook_ids": []
}
```


### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| alert_id | path | integer | true | アラート ID |
| X-Auth-Token | header | string | true | 承認トークン |
| body | body | *AlertPayload* | true | なし |

### 応答例

応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ステータス 200 | None |

レスポンススキーマ

> **注釈**
>
> この操作には認証が必要ありません

### delete_*alert_config_global*{alert_id}

コードサンプル

```
#  You can also use wget

curl -X DELETE https://PEM-SERVER-IP/pem/api/v17/alert/config/global/{alert_id} \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  X-Auth-Token: string
}

r = requests.delete(https://PEM-SERVER-IP/pem/api/v17/alert/config/global/{alert_id},␣
↪headers = headers)

print(r.json())
```

```
const headers = {
  "X-Auth-Token": "string",
};

fetch("https://PEM-SERVER-IP/pem/api/v17/alert/config/global/{alert_id}", {
  method: "DELETE",

  headers: headers,
})
  .then(function (res) {
    return res.json();
```

(次のページに続く)

(前のページからの続き)

```
  })
  .then(function (body) {
    console.log(body);
  });
```

`DELETE /alert/config/global/{alert_id}`

`Alert` オブジェクトを削除します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| alert_id | path | integer | true | アラート ID |
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 削除が成功しました。 | None |

### レスポンススキーマ

**注釈**

この操作には認証が必要ありません

## get_*alert_config_agent*{agent_id}

### コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/alert/config/agent/{agent_id} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/alert/config/agent/{agent_id},␣
↪headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch("https://PEM-SERVER-IP/pem/api/v17/alert/config/agent/{agent_id}", {
  method: "GET",

  headers: headers,
})
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

`GET /alert/config/agent/{agent_id}`

agent_id でエージェントレベルの `Alerts` を取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| agent_id | path | integer | true | エージェント ID |
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例

```
[
  {
    "id": 0,
    "alert_name": "string",
    "alert_template": 0,
```

(次のページに続く)

(前のページからの続き)


```
    "description": "string",
    "enabled": true,
    "history_retention": 0,
    "frequency_min": 0,
    "auto_created": true,
    "template_id": "string",
    "params": [],
    "param_names": ["string"],
    "param_types": "string",
    "params_units": ["string"],
    "threshold_unit": "string",
    "operator": "string",
    "low_threshold_value": 0,
    "medium_threshold_value": 0,
    "high_threshold_value": 0,
    "send_email": true,
    "all_alert_enable": true,
    "email_group_id": 0,
    "low_alert_enable": true,
    "low_email_group_id": 0,
    "med_alert_enable": true,
    "med_email_group_id": 0,
    "high_alert_enable": true,
    "high_email_group_id": 0,
    "send_trap": true,
    "snmp_trap_version": 0,
    "low_send_trap": true,
    "med_send_trap": true,
    "high_send_trap": true,
    "cleared_alert_enable": true,
    "execute_script": true,
    "execute_script_on_clear": true,
    "execute_script_on_pem_server": true,
    "script_code": "string",
    "submit_to_nagios": true,
    "override_default_config": true,
    "send_notification": true,
    "low_webhook_ids": [0],
    "med_webhook_ids": [0],
    "high_webhook_ids": [0],
    "cleared_webhook_ids": [0]
  }
]
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | Inline |

### レスポンススキーマ

ステータスコード **200**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| _anonymous_ | *Platform Compatibility* ] | false | none | なし |
| » id | integer | false | none | アラート ID。注このプロパティは、POST 要求には適用されません。 |
| » alert_name | string | false | none | アラートの名前。注 このプロパティは、POST 要求の必須パラメーターです。 |
| » alert_template | integer | false | none | アラートテンプレートの ID。このプロパティは、正の整数値である必要があります。注これはPOST 要求の必須パラメーターです。 |
| » description | string | false | none | なし |
| » enabled | boolean | false | none | このアラートが有効/無効かどうかを判断します。注 このプロパティは、POST 要求の必須パラメーターです。 |
| » history_retention | integer | false | none | このアラートの履歴を保持する日数を指定します。指定できる値は 1 ～99999 です。 注このプロパティは、POST 要求の必須パラメーターです。 |

次のページに続く

表 263 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » frequency_min | integer | false | none | PEM データベースサーバーでアラートを確認する頻度を分単位で定義します。指定できる値は 1〜65534 です。 注このプロパティは、POST 要求の必須パラメーターです。 |
| » auto_created | boolean | false | none | アラートを自動作成するかどうかを指定します。 |
| » template_id | string | false | none | アラートテンプレートを定義します。 |
| » params | *Platform Compatibility* ] | false | none | アラートを確認するためにアラートテンプレートが使用するパラメーター/値のペアのリスト。注アラートテンプレートには、all パラメーターを定義する必要があります。 |
| » paramname | string | false | none | パラメーターの名前。注このプロパティは、アラートテンプレートパラメーターリストに存在する必要があります。例 |
| » paramvalue | string | false | none | 指定されたパラメーターの値 |
| » param_names | [string] | false | none | パラメーターの名前を定義します。 |
| » param_types | string | false | none | パラメーターのタイプを定義します。例 INTEGER/STRING.. |
| » params_units | [string] | false | none | パラメーターに使用される単位を定義します。例 'DAYS'、'True or False'、'Hours'、'Minutes' など |
| » threshold_unit | string | false | none | MB/kB/秒/日/... などの、しきい値の計算に使用される単位 この値は、GUI でのみ使用されます。 |

次のページに続く

表 263 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » operator | string | false | none | しきい値をチェックするための演算子を定義します。有効な文字列は>"および"<"です。注このプロパティは、POST要求の必須パラメーターです。 |
| » low_threshold_value | number | false | none | 低アラートをトリガーするためのしきい値。このプロパティには、有効な整数/浮動小数点値を指定できます。注これはPOST要求の必須パラメーターです。 |
| » medium_threshold_value | number | false | none | 中アラートをトリガーするためのしきい値。このプロパティには、有効な整数/浮動小数点値を指定できます。注これはPOST要求の必須パラメーターです。 |
| » high_threshold_value | number | false | none | 高度なアラートをトリガーするためのしきい値。任意の有効な整数/浮動小数点値を指定できます。注これはPOST要求の必須パラメーターです。 |
| » send_email | boolean | false | none | アラートの状態が変更されたときに電子メールを送信するかどうかを決定します。 |
| » all_alert_enable | boolean | false | none | アラートの状態が変更されたときに、パラメーター 'email_group_id' で指定されたグループにメール通知を送信するかどうかを決定します。 |
| » email_group_id | integer | false | none | アラートの状態が変更されたときに電子メール通知の送信先の電子メールグループを定義します。 |

次のページに続く

表 263 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » low_alert_enable | boolean | false | none | アラートの状態が LOW に設定されているときに、パラメーター 'low_email_group_id' で指定されたグループにメール通知を送信するかどうかを決定します。 |
| » low_email_group_id | integer | false | none | アラートの状態が LOW に変更されたときに電子メール通知の送信先の電子メールグループを定義します。 |
| » med_alert_enable | boolean | false | none | アラートの状態が MEDIUM に設定されている場合、パラメーター 'medium_email_group_id' で指定されたグループにメール通知を送信するかどうかを決定します。 |
| » med_email_group_id | integer | false | none | アラートの状態が MEDIUM に変更されたときにメール通知の送信先のメールグループを定義します。 |
| » high_alert_enable | boolean | false | none | アラートの状態が HIGH に設定されているときに、パラメーター 'high_email_group_id' で指定されたグループに電子メール通知を送信するかどうかを決定します。 |
| » high_email_group_id | integer | false | none | アラートの状態が HIGH に変更されたときに電子メール通知の送信先の電子メールグループを定義します。 |
| » send_trap | boolean | false | none | アラートの状態が変更されたときに SNMP トラップ通知を送信するかどうかを決定します。 |

次のページに続く

表 263 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » snmp_trap_version | integer | false | none | SNMP トラップバージョンを定義します。サポートされているSNMP トラップバージョンは 1、2、または 3 です |
| » low_send_trap | boolean | false | none | アラートの状態が LOW に変更されたときに SNMP トラップ通知を送信するかどうかを決定します。 |
| » med_send_trap | boolean | false | none | アラートの状態が MEDIUM に変更されたときに SNMP トラップ通知を送信するかどうかを決定します。 |
| » high_send_trap | boolean | false | none | アラートの状態が HIGH に変わったときに SNMP トラップ通知を送信するかどうかを決定します。 |
| » cleared_alert_enable | boolean | false | none | ' クリアアラート' の通知をミュートするかどうかを決定します。 |
| » execute_script | boolean | false | none | アラートの状態が LOW/MEDIUM/HIGH に変更された場合に、提供されたスクリプトを実行するかどうかを決定します。 |
| » execute_script_on_clear | boolean | false | none | アラートの状態がクリアされた場合に、提供されたスクリプトを実行するかどうかを決定します。 |
| » execute_script_on_pem_serve | boolean | false | none | 提供されたスクリプトを PEM ホストまたはエージェントがオブジェクトを監視しているホストで実行するかどうかを決定します。 |

次のページに続く

表 263 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » script_code | string | false | none | 実行するシェルまたはバッチスクリプト。詳細については、ドキュメントをご覧ください。 |
| » submit_to_nagios | boolean | false | none | アラートの状態が変更されたときに nagios に通知を送信するかどうかを決定します。 |
| » override_default_config | boolean | false | none | デフォルトの Webhook 構成をオーバーライドするかどうかを決定します。 |
| » send_notification | boolean | false | none | Webhook 送信通知を有効にするかどうかを決定します。 |
| » low_webhook_ids | [integer] | false | none | 低アラートが構成されている Webhook ID |
| » med_webhook_ids | [integer] | false | none | 中アラートが構成されている Webhook ID |
| » high_webhook_ids | [integer] | false | none | 高アラートが構成されている Webhook ID |
| » cleared_webhook_ids | [integer] | false | none | クリアされたアラートが構成されている Webhook ID |

### 列挙値

| Property | Value |
|---|---|
| paramname | Exclude tables smaller than |
| paramname | mount point |
| paramname | Host IP Address |
| paramname | Time interval to monitor |
| paramname | Time since idle |
| paramname | Per CPU utilization (K%) |

**注釈**

この操作には認証が必要ありません

### post_*alert_config_agent*{agent_id}

コードサンプル

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v17/alert/config/agent/{agent_id} \
  -H Content-Type: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Content-Type: application/json,
  X-Auth-Token: string
}

r = requests.post(https://PEM-SERVER-IP/pem/api/v17/alert/config/agent/{agent_id},␣
↪headers = headers)

print(r.json())
```

```
const inputBody = {
  "alert_name": "string",
  "alert_template": 0,
  "description": "string",
  "enabled": true,
  "history_retention": 0,
  "frequency_min": 0,
  "operator": "string",
  "low_threshold_value": 0,
  "medium_threshold_value": 0,
  "high_threshold_value": 0,
  "all_alert_enable": true,
  "email_group_id": 1,
  "low_alert_enable": true,
  "low_email_group_id": 1,
  "med_alert_enable": true,
  "med_email_group_id": 1,
  "high_alert_enable": true,
  "high_email_group_id": 1,
  "send_trap": false,
  "snmp_trap_version": 0,
  "low_send_trap": true,
  "med_send_trap": true,
  "high_send_trap": true,
  "cleared_alert_enable": true,
```

(次のページに続く)

(前のページからの続き)

```
  "params": [],
  "execute_script": true,
  "execute_script_on_clear": true,
  "execute_script_on_pem_server": true,
  "script_code": "string",
  "submit_to_nagios": true,
  "override_default_config": true,
  "send_notification": false,
  "low_webhook_ids": [],
  "med_webhook_ids": [],
  "high_webhook_ids": [],
  "cleared_webhook_ids": []
};
const headers = {
  Content-Type:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v17/alert/config/agent/{agent_id},
{
  method: POST,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

POST /alert/config/agent/{agent_id}

エージェントレベルの Alert を作成します。

### ボディパラメーター

```
{
  "alert_name": "string",
  "alert_template": 0,
  "description": "string",
  "enabled": true,
  "history_retention": 0,
  "frequency_min": 0,
  "operator": "string",
  "low_threshold_value": 0,
  "medium_threshold_value": 0,
```

(次のページに続く)

(前のページからの続き)


```
  "high_threshold_value": 0,
  "all_alert_enable": true,
  "email_group_id": 1,
  "low_alert_enable": true,
  "low_email_group_id": 1,
  "med_alert_enable": true,
  "med_email_group_id": 1,
  "high_alert_enable": true,
  "high_email_group_id": 1,
  "send_trap": false,
  "snmp_trap_version": 0,
  "low_send_trap": true,
  "med_send_trap": true,
  "high_send_trap": true,
  "cleared_alert_enable": true,
  "params": [],
  "execute_script": true,
  "execute_script_on_clear": true,
  "execute_script_on_pem_server": true,
  "script_code": "string",
  "submit_to_nagios": true,
  "override_default_config": true,
  "send_notification": false,
  "low_webhook_ids": [],
  "med_webhook_ids": [],
  "high_webhook_ids": [],
  "cleared_webhook_ids": []
}
```


### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| agent_id | path | integer | true | エージェント ID |
| X-Auth-Token | header | string | true | 承認トークン |
| body | body | *AlertPayload* | true | なし |

### 応答例

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ステータス 200 | None |

### レスポンススキーマ

> **注釈**
>
> この操作には認証が必要ありません

### get_*alert_config_agent*{agent_id}_{alert_id}

#### コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/alert/config/agent/{agent_id}/{alert_id}
↪ \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/alert/config/agent/{agent_id}/
↪{alert_id}, headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch(
  "https://PEM-SERVER-IP/pem/api/v17/alert/config/agent/{agent_id}/{alert_id}",
  {
```

(次のページに続く)

(前のページからの続き)

```
    method: "GET",

    headers: headers,
  },
)
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

`GET /alert/config/agent/{agent_id}/{alert_id}`

agent_id および alert_id でエージェントレベルの `Alerts` を取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| agent_id | path | integer | true | エージェント ID |
| alert_id | path | integer | true | アラート ID |
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例

```
{
  "id": 0,
  "alert_name": "string",
  "alert_template": 0,
  "description": "string",
  "enabled": true,
  "history_retention": 0,
  "frequency_min": 0,
  "auto_created": true,
  "template_id": "string",
  "params": [],
  "param_names": ["string"],
  "param_types": "string",
  "params_units": ["string"],
  "threshold_unit": "string",
  "operator": "string",
  "low_threshold_value": 0,
  "medium_threshold_value": 0,
  "high_threshold_value": 0,
```

(次のページに続く)

(前のページからの続き)


```
  "send_email": true,
  "all_alert_enable": true,
  "email_group_id": 0,
  "low_alert_enable": true,
  "low_email_group_id": 0,
  "med_alert_enable": true,
  "med_email_group_id": 0,
  "high_alert_enable": true,
  "high_email_group_id": 0,
  "send_trap": true,
  "snmp_trap_version": 0,
  "low_send_trap": true,
  "med_send_trap": true,
  "high_send_trap": true,
  "cleared_alert_enable": true,
  "execute_script": true,
  "execute_script_on_clear": true,
  "execute_script_on_pem_server": true,
  "script_code": "string",
  "submit_to_nagios": true,
  "override_default_config": true,
  "send_notification": true,
  "low_webhook_ids": [0],
  "med_webhook_ids": [0],
  "high_webhook_ids": [0],
  "cleared_webhook_ids": [0]
}
```


応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | アラートスレッドについて |

**注釈**

この操作には認証が必要ありません

### put_*alert_config_agent*{agent_id}_{alert_id}

コードサンプル

```
#  You can also use wget

curl -X PUT https://PEM-SERVER-IP/pem/api/v17/alert/config/agent/{agent_id}/{alert_id}
↪ \
  -H Content-Type: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Content-Type: application/json,
  X-Auth-Token: string
}

r = requests.put(https://PEM-SERVER-IP/pem/api/v17/alert/config/agent/{agent_id}/
↪{alert_id}, headers = headers)

print(r.json())
```

```
const inputBody = {
  "alert_name": "string",
  "alert_template": 0,
  "description": "string",
  "enabled": true,
  "history_retention": 0,
  "frequency_min": 0,
  "operator": "string",
  "low_threshold_value": 0,
  "medium_threshold_value": 0,
  "high_threshold_value": 0,
  "all_alert_enable": true,
  "email_group_id": 1,
  "low_alert_enable": true,
  "low_email_group_id": 1,
  "med_alert_enable": true,
  "med_email_group_id": 1,
  "high_alert_enable": true,
  "high_email_group_id": 1,
  "send_trap": false,
  "snmp_trap_version": 0,
  "low_send_trap": true,
  "med_send_trap": true,
  "high_send_trap": true,
```

(次のページに続く)

(前のページからの続き)

```
  "cleared_alert_enable": true,
  "params": [],
  "execute_script": true,
  "execute_script_on_clear": true,
  "execute_script_on_pem_server": true,
  "script_code": "string",
  "submit_to_nagios": true,
  "override_default_config": true,
  "send_notification": false,
  "low_webhook_ids": [],
  "med_webhook_ids": [],
  "high_webhook_ids": [],
  "cleared_webhook_ids": []
};
const headers = {
  Content-Type:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v17/alert/config/agent/{agent_id}/{alert_id},
{
  method: PUT,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

PUT /alert/config/agent/{agent_id}/{alert_id}

エージェントレベルの Alert を更新します。

### ボディパラメーター

```
{
  "alert_name": "string",
  "alert_template": 0,
  "description": "string",
  "enabled": true,
  "history_retention": 0,
  "frequency_min": 0,
  "operator": "string",
  "low_threshold_value": 0,
```

(次のページに続く)

(前のページからの続き)


```
  "medium_threshold_value": 0,
  "high_threshold_value": 0,
  "all_alert_enable": true,
  "email_group_id": 1,
  "low_alert_enable": true,
  "low_email_group_id": 1,
  "med_alert_enable": true,
  "med_email_group_id": 1,
  "high_alert_enable": true,
  "high_email_group_id": 1,
  "send_trap": false,
  "snmp_trap_version": 0,
  "low_send_trap": true,
  "med_send_trap": true,
  "high_send_trap": true,
  "cleared_alert_enable": true,
  "params": [],
  "execute_script": true,
  "execute_script_on_clear": true,
  "execute_script_on_pem_server": true,
  "script_code": "string",
  "submit_to_nagios": true,
  "override_default_config": true,
  "send_notification": false,
  "low_webhook_ids": [],
  "med_webhook_ids": [],
  "high_webhook_ids": [],
  "cleared_webhook_ids": []
}
```


### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| agent_id | path | integer | true | エージェント ID |
| alert_id | path | integer | true | アラート ID |
| X-Auth-Token | header | string | true | 承認トークン |
| body | body | *AlertPayload* | true | なし |

### 応答例

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ステータス 200 | None |

### レスポンススキーマ

> **注釈**
>
> この操作には認証が必要ありません

## delete_*alert_config_agent*{agent_id}_{alert_id}

### コードサンプル

```
#  You can also use wget

curl -X DELETE https://PEM-SERVER-IP/pem/api/v17/alert/config/agent/{agent_id}/{alert_
↪id} \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  X-Auth-Token: string
}

r = requests.delete(https://PEM-SERVER-IP/pem/api/v17/alert/config/agent/{agent_id}/
↪{alert_id}, headers = headers)

print(r.json())
```

```
const headers = {
  "X-Auth-Token": "string",
};

fetch(
  "https://PEM-SERVER-IP/pem/api/v17/alert/config/agent/{agent_id}/{alert_id}",
  {
    method: "DELETE",

    headers: headers,
```

(次のページに続く)

(前のページからの続き)

```
  },
)
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

`DELETE /alert/config/agent/{agent_id}/{alert_id}`

`Alert` オブジェクトを削除します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| agent_id | path | integer | true | エージェント ID |
| alert_id | path | integer | true | アラート ID |
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 削除が成功しました。 | None |

### レスポンススキーマ

**注釈**

この操作には認証が必要ありません

## get_*alert_config_server*{server_id}

### コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id} \
```

(次のページに続く)

(前のページからの続き)

```
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id},␣
↪headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch("https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}", {
  method: "GET",

  headers: headers,
})
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /alert/config/server/{server_id}

server_id によりサーバーレベルの Alerts を取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例

```
[
  {
    "id": 0,
    "alert_name": "string",
    "alert_template": 0,
    "description": "string",
    "enabled": true,
    "history_retention": 0,
    "frequency_min": 0,
    "auto_created": true,
    "template_id": "string",
    "params": [],
    "param_names": ["string"],
    "param_types": "string",
    "params_units": ["string"],
    "threshold_unit": "string",
    "operator": "string",
    "low_threshold_value": 0,
    "medium_threshold_value": 0,
    "high_threshold_value": 0,
    "send_email": true,
    "all_alert_enable": true,
    "email_group_id": 0,
    "low_alert_enable": true,
    "low_email_group_id": 0,
    "med_alert_enable": true,
    "med_email_group_id": 0,
    "high_alert_enable": true,
    "high_email_group_id": 0,
    "send_trap": true,
    "snmp_trap_version": 0,
    "low_send_trap": true,
    "med_send_trap": true,
    "high_send_trap": true,
    "cleared_alert_enable": true,
    "execute_script": true,
    "execute_script_on_clear": true,
    "execute_script_on_pem_server": true,
    "script_code": "string",
    "submit_to_nagios": true,
    "override_default_config": true,
    "send_notification": true,
    "low_webhook_ids": [0],
    "med_webhook_ids": [0],
    "high_webhook_ids": [0],
```


(次のページに続く)

(前のページからの続き)

```
    "cleared_webhook_ids": [0]
  }
]
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | Inline |

### レスポンススキーマ

ステータスコード **200**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| _anonymous_ | *Platform Compatibility* ] | false | none | なし |
| » id | integer | false | none | アラート ID。注このプロパティは、POST 要求には適用されません。 |
| » alert_name | string | false | none | アラートの名前。注 このプロパティは、POST 要求の必須パラメーターです。 |
| » alert_template | integer | false | none | アラートテンプレートの ID。このプロパティは、正の整数値である必要があります。注これはPOST 要求の必須パラメーターです。 |
| » description | string | false | none | なし |
| » enabled | boolean | false | none | このアラートが有効/無効かどうかを判断します。注 このプロパティは、POST 要求の必須パラメーターです。 |

次のページに続く

表 275 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » history_retention | integer | false | none | このアラートの履歴を保持する日数を指定します。指定できる値は 1 〜99999 です。 注このプロパティは、POST 要求の必須パラメーターです。 |
| » frequency_min | integer | false | none | PEM データベースサーバーでアラートを確認する頻度を分単位で定義します。指定できる値は 1 〜65534 です。 注このプロパティは、POST 要求の必須パラメーターです。 |
| » auto_created | boolean | false | none | アラートを自動作成するかどうかを指定します。 |
| » template_id | string | false | none | アラートテンプレートを定義します。 |
| » params | *Platform Compatibility* ] | false | none | アラートを確認するためにアラートテンプレートが使用するパラメーター/値のペアのリスト。注アラートテンプレートには、all パラメーターを定義する必要があります。 |
| » paramname | string | false | none | パラメーターの名前。注このプロパティは、アラートテンプレートパラメーターリストに存在する必要があります。例 |
| » paramvalue | string | false | none | 指定されたパラメーターの値 |
| » param_names | [string] | false | none | パラメーターの名前を定義します。 |
| » param_types | string | false | none | パラメーターのタイプを定義します。例 INTEGER/STRING.. |
| » params_units | [string] | false | none | パラメーターに使用される単位を定義します。例 'DAYS'、'True or False'、'Hours'、'Minutes' など |

次のページに続く

表 275 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » threshold_unit | string | false | none | MB/kB/秒/日/... などの、しきい値の計算に使用される単位 この値は、GUI でのみ使用されます。 |
| » operator | string | false | none | しきい値をチェックするための演算子を定義します。有効な文字列は>"および"<"です。注 このプロパティは、POST 要求の必須パラメーターです。 |
| » low_threshold_value | number | false | none | 低アラートをトリガーするためのしきい値。このプロパティには、有効な整数/浮動小数点値を指定できます。注これは POST 要求の必須パラメーターです。 |
| » medium_threshold_value | number | false | none | 中アラートをトリガーするためのしきい値。このプロパティには、有効な整数/浮動小数点値を指定できます。注これは POST 要求の必須パラメーターです。 |
| » high_threshold_value | number | false | none | 高度なアラートをトリガーするためのしきい値。任意の有効な整数/浮動小数点値を指定できます。注これは POST 要求の必須パラメーターです。 |
| » send_email | boolean | false | none | アラートの状態が変更されたときに電子メールを送信するかどうかを決定します。 |
| » all_alert_enable | boolean | false | none | アラートの状態が変更されたときに、パラメーター 'email_group_id' で指定されたグループにメール通知を送信するかどうかを決定します。 |

次のページに続く

表 275 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » email_group_id | integer | false | none | アラートの状態が変更されたときに電子メール通知の送信先の電子メールグループを定義します。 |
| » low_alert_enable | boolean | false | none | アラートの状態が LOW に設定されているときに、パラメーター 'low_email_group_id' で指定されたグループにメール通知を送信するかどうかを決定します。 |
| » low_email_group_id | integer | false | none | アラートの状態が LOW に変更されたときに電子メール通知の送信先の電子メールグループを定義します。 |
| » med_alert_enable | boolean | false | none | アラートの状態が MEDIUM に設定されている場合、パラメーター 'medium_email_group_id' で指定されたグループにメール通知を送信するかどうかを決定します。 |
| » med_email_group_id | integer | false | none | アラートの状態が MEDIUM に変更されたときにメール通知の送信先のメールグループを定義します。 |
| » high_alert_enable | boolean | false | none | アラートの状態が HIGH に設定されているときに、パラメーター 'high_email_group_id' で指定されたグループに電子メール通知を送信するかどうかを決定します。 |
| » high_email_group_id | integer | false | none | アラートの状態が HIGH に変更されたときに電子メール通知の送信先の電子メールグループを定義します。 |

次のページに続く

表 275 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » send_trap | boolean | false | none | アラートの状態が変更されたときに SNMP トラップ通知を送信するかどうかを決定します。 |
| » snmp_trap_version | integer | false | none | SNMP トラップバージョンを定義します。サポートされている SNMP トラップバージョンは 1、2、または 3 です |
| » low_send_trap | boolean | false | none | アラートの状態が LOW に変更されたときに SNMP トラップ通知を送信するかどうかを決定します。 |
| » med_send_trap | boolean | false | none | アラートの状態が MEDIUM に変更されたときに SNMP トラップ通知を送信するかどうかを決定します。 |
| » high_send_trap | boolean | false | none | アラートの状態が HIGH に変わったときに SNMP トラップ通知を送信するかどうかを決定します。 |
| » cleared_alert_enable | boolean | false | none | 'クリアアラート' の通知をミュートするかどうかを決定します。 |
| » execute_script | boolean | false | none | アラートの状態が LOW/MEDIUM/HIGH に変更された場合に、提供されたスクリプトを実行するかどうかを決定します。 |
| » execute_script_on_clear | boolean | false | none | アラートの状態がクリアされた場合に、提供されたスクリプトを実行するかどうかを決定します。 |

次のページに続く

表 275 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » execute_script_on_pem_serve | boolean | false | none | 提供されたスクリプトをPEM ホストまたはエージェントがオブジェクトを監視しているホストで実行するかどうかを決定します。 |
| » script_code | string | false | none | 実行するシェルまたはバッチスクリプト。詳細については、ドキュメントをご覧ください。 |
| » submit_to_nagios | boolean | false | none | アラートの状態が変更されたときに nagios に通知を送信するかどうかを決定します。 |
| » override_default_config | boolean | false | none | デフォルトの Webhook 構成をオーバーライドするかどうかを決定します。 |
| » send_notification | boolean | false | none | Webhook 送信通知を有効にするかどうかを決定します。 |
| » low_webhook_ids | [integer] | false | none | 低アラートが構成されている Webhook ID |
| » med_webhook_ids | [integer] | false | none | 中アラートが構成されている Webhook ID |
| » high_webhook_ids | [integer] | false | none | 高アラートが構成されている Webhook ID |
| » cleared_webhook_ids | [integer] | false | none | クリアされたアラートが構成されている Webhook ID |

### 列挙値

| Property | Value |
|---|---|
| paramname | Exclude tables smaller than |
| paramname | mount point |
| paramname | Host IP Address |
| paramname | Time interval to monitor |
| paramname | Time since idle |
| paramname | Per CPU utilization (K%) |

> **注釈**
>
> この操作には認証が必要ありません

### post_alert_config_server{server_id}

#### コードサンプル

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id} \
  -H Content-Type: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Content-Type: application/json,
  X-Auth-Token: string
}

r = requests.post(https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id},␣
↪headers = headers)

print(r.json())
```

```
const inputBody = {
  "alert_name": "string",
  "alert_template": 0,
  "description": "string",
  "enabled": true,
  "history_retention": 0,
  "frequency_min": 0,
  "operator": "string",
  "low_threshold_value": 0,
  "medium_threshold_value": 0,
  "high_threshold_value": 0,
  "all_alert_enable": true,
  "email_group_id": 1,
  "low_alert_enable": true,
  "low_email_group_id": 1,
  "med_alert_enable": true,
  "med_email_group_id": 1,
  "high_alert_enable": true,
  "high_email_group_id": 1,
  "send_trap": false,
```

(次のページに続く)

(前のページからの続き)

```
  "snmp_trap_version": 0,
  "low_send_trap": true,
  "med_send_trap": true,
  "high_send_trap": true,
  "cleared_alert_enable": true,
  "params": [],
  "execute_script": true,
  "execute_script_on_clear": true,
  "execute_script_on_pem_server": true,
  "script_code": "string",
  "submit_to_nagios": true,
  "override_default_config": true,
  "send_notification": false,
  "low_webhook_ids": [],
  "med_webhook_ids": [],
  "high_webhook_ids": [],
  "cleared_webhook_ids": []
};
const headers = {
  Content-Type:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id},
{
  method: POST,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

POST /alert/config/server/{server_id}

サーバーレベルの Alert を作成します。

### ボディパラメーター

```
{
  "alert_name": "string",
  "alert_template": 0,
  "description": "string",
  "enabled": true,
```

(次のページに続く)

(前のページからの続き)


```
  "history_retention": 0,
  "frequency_min": 0,
  "operator": "string",
  "low_threshold_value": 0,
  "medium_threshold_value": 0,
  "high_threshold_value": 0,
  "all_alert_enable": true,
  "email_group_id": 1,
  "low_alert_enable": true,
  "low_email_group_id": 1,
  "med_alert_enable": true,
  "med_email_group_id": 1,
  "high_alert_enable": true,
  "high_email_group_id": 1,
  "send_trap": false,
  "snmp_trap_version": 0,
  "low_send_trap": true,
  "med_send_trap": true,
  "high_send_trap": true,
  "cleared_alert_enable": true,
  "params": [],
  "execute_script": true,
  "execute_script_on_clear": true,
  "execute_script_on_pem_server": true,
  "script_code": "string",
  "submit_to_nagios": true,
  "override_default_config": true,
  "send_notification": false,
  "low_webhook_ids": [],
  "med_webhook_ids": [],
  "high_webhook_ids": [],
  "cleared_webhook_ids": []
}
```


### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| X-Auth-Token | header | string | true | 承認トークン |
| body | body | *AlertPayload* | true | なし |

### 応答例

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ステータス 200 | None |

### レスポンススキーマ

> **注釈**
>
> この操作には認証が必要ありません

## get_*alert_config_server*{server_id}_{alert_id}

### コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/{alert_
↪id} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/
↪{alert_id}, headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch(
  "https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/{alert_id}",
  {
```

(次のページに続く)

(前のページからの続き)

```
    method: "GET",

    headers: headers,
  },
)
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

`GET /alert/config/server/{server_id}/{alert_id}`

server_id および alert_id によってサーバーレベルの `Alerts` を取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| alert_id | path | integer | true | アラート ID |
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例

```
{
  "id": 0,
  "alert_name": "string",
  "alert_template": 0,
  "description": "string",
  "enabled": true,
  "history_retention": 0,
  "frequency_min": 0,
  "auto_created": true,
  "template_id": "string",
  "params": [],
  "param_names": ["string"],
  "param_types": "string",
  "params_units": ["string"],
  "threshold_unit": "string",
  "operator": "string",
  "low_threshold_value": 0,
  "medium_threshold_value": 0,
  "high_threshold_value": 0,
```

(次のページに続く)

(前のページからの続き)


```
  "send_email": true,
  "all_alert_enable": true,
  "email_group_id": 0,
  "low_alert_enable": true,
  "low_email_group_id": 0,
  "med_alert_enable": true,
  "med_email_group_id": 0,
  "high_alert_enable": true,
  "high_email_group_id": 0,
  "send_trap": true,
  "snmp_trap_version": 0,
  "low_send_trap": true,
  "med_send_trap": true,
  "high_send_trap": true,
  "cleared_alert_enable": true,
  "execute_script": true,
  "execute_script_on_clear": true,
  "execute_script_on_pem_server": true,
  "script_code": "string",
  "submit_to_nagios": true,
  "override_default_config": true,
  "send_notification": true,
  "low_webhook_ids": [0],
  "med_webhook_ids": [0],
  "high_webhook_ids": [0],
  "cleared_webhook_ids": [0]
}
```


応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | アラートスレッドについて |

**注釈**

この操作には認証が必要ありません

### put_*alert_config_server*{server_id}_{alert_id}

コードサンプル

```
#  You can also use wget

curl -X PUT https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/{alert_
↪id} \
  -H Content-Type: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Content-Type: application/json,
  X-Auth-Token: string
}

r = requests.put(https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/
↪{alert_id}, headers = headers)

print(r.json())
```

```
const inputBody = {
  "alert_name": "string",
  "alert_template": 0,
  "description": "string",
  "enabled": true,
  "history_retention": 0,
  "frequency_min": 0,
  "operator": "string",
  "low_threshold_value": 0,
  "medium_threshold_value": 0,
  "high_threshold_value": 0,
  "all_alert_enable": true,
  "email_group_id": 1,
  "low_alert_enable": true,
  "low_email_group_id": 1,
  "med_alert_enable": true,
  "med_email_group_id": 1,
  "high_alert_enable": true,
  "high_email_group_id": 1,
  "send_trap": false,
  "snmp_trap_version": 0,
  "low_send_trap": true,
  "med_send_trap": true,
  "high_send_trap": true,
```

(次のページに続く)

(前のページからの続き)

```
  "cleared_alert_enable": true,
  "params": [],
  "execute_script": true,
  "execute_script_on_clear": true,
  "execute_script_on_pem_server": true,
  "script_code": "string",
  "submit_to_nagios": true,
  "override_default_config": true,
  "send_notification": false,
  "low_webhook_ids": [],
  "med_webhook_ids": [],
  "high_webhook_ids": [],
  "cleared_webhook_ids": []
};
const headers = {
  Content-Type:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/{alert_id},
{
  method: PUT,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

PUT /alert/config/server/{server_id}/{alert_id}

サーバーレベルの Alert を更新します。

**ボディパラメーター**

```
{
  "alert_name": "string",
  "alert_template": 0,
  "description": "string",
  "enabled": true,
  "history_retention": 0,
  "frequency_min": 0,
  "operator": "string",
  "low_threshold_value": 0,
```

(次のページに続く)

(前のページからの続き)


```
  "medium_threshold_value": 0,
  "high_threshold_value": 0,
  "all_alert_enable": true,
  "email_group_id": 1,
  "low_alert_enable": true,
  "low_email_group_id": 1,
  "med_alert_enable": true,
  "med_email_group_id": 1,
  "high_alert_enable": true,
  "high_email_group_id": 1,
  "send_trap": false,
  "snmp_trap_version": 0,
  "low_send_trap": true,
  "med_send_trap": true,
  "high_send_trap": true,
  "cleared_alert_enable": true,
  "params": [],
  "execute_script": true,
  "execute_script_on_clear": true,
  "execute_script_on_pem_server": true,
  "script_code": "string",
  "submit_to_nagios": true,
  "override_default_config": true,
  "send_notification": false,
  "low_webhook_ids": [],
  "med_webhook_ids": [],
  "high_webhook_ids": [],
  "cleared_webhook_ids": []
}
```


### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| alert_id | path | integer | true | アラート ID |
| X-Auth-Token | header | string | true | 承認トークン |
| body | body | *AlertPayload* | true | なし |

### 応答例

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ステータス 200 | None |

### レスポンススキーマ

> **注釈**
>
> この操作には認証が必要ありません

## delete_*alert_config_server*{server_id}_{alert_id}

### コードサンプル

```
#  You can also use wget

curl -X DELETE https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/
↪{alert_id} \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  X-Auth-Token: string
}

r = requests.delete(https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/
↪{alert_id}, headers = headers)

print(r.json())
```

```
const headers = {
  "X-Auth-Token": "string",
};

fetch(
  "https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/{alert_id}",
  {
    method: "DELETE",

    headers: headers,
```

(次のページに続く)

(前のページからの続き)

```
  },
)
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

`DELETE /alert/config/server/{server_id}/{alert_id}`

`Alert` オブジェクトを削除します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| alert_id | path | integer | true | アラート ID |
| X-Auth-Token | header | string | true | 承認トークン |

**応答例**

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 削除が成功しました。 | None |

### レスポンススキーマ

**注釈**

この操作には認証が必要ありません

## get_*alert_config_server*{server_id}*database*{database_name}

**コードサンプル**

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/
```

(次のページに続く)

(前のページからの続き)

```
↪database/{database_name} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/
↪database/{database_name}, headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch(
  "https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/database/
↪{database_name}",
  {
    method: "GET",

    headers: headers,
  },
)
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /alert/config/server/{server_id}/database/{database_name}

server_id および database_name により、データベースレベルの Alerts を取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| database_name | path | string | true | データベース名 |
| X-Auth-Token | header | string | true | 承認トークン |

応答例


```
[
  {
    "id": 0,
    "alert_name": "string",
    "alert_template": 0,
    "description": "string",
    "enabled": true,
    "history_retention": 0,
    "frequency_min": 0,
    "auto_created": true,
    "template_id": "string",
    "params": [],
    "param_names": ["string"],
    "param_types": "string",
    "params_units": ["string"],
    "threshold_unit": "string",
    "operator": "string",
    "low_threshold_value": 0,
    "medium_threshold_value": 0,
    "high_threshold_value": 0,
    "send_email": true,
    "all_alert_enable": true,
    "email_group_id": 0,
    "low_alert_enable": true,
    "low_email_group_id": 0,
    "med_alert_enable": true,
    "med_email_group_id": 0,
    "high_alert_enable": true,
    "high_email_group_id": 0,
    "send_trap": true,
    "snmp_trap_version": 0,
    "low_send_trap": true,
    "med_send_trap": true,
    "high_send_trap": true,
    "cleared_alert_enable": true,
    "execute_script": true,
    "execute_script_on_clear": true,
    "execute_script_on_pem_server": true,
```


(次のページに続く)

(前のページからの続き)

```
    "script_code": "string",
    "submit_to_nagios": true,
    "override_default_config": true,
    "send_notification": true,
    "low_webhook_ids": [0],
    "med_webhook_ids": [0],
    "high_webhook_ids": [0],
    "cleared_webhook_ids": [0]
  }
]
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | Inline |

### レスポンススキーマ

ステータスコード **200**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| _anonymous_ | *Platform Compatibility* ] | false | none | なし |
| » id | integer | false | none | アラート ID。注このプロパティは、POST 要求には適用されません。 |
| » alert_name | string | false | none | アラートの名前。注 このプロパティは、POST 要求の必須パラメーターです。 |
| » alert_template | integer | false | none | アラートテンプレートの ID。このプロパティは、正の整数値である必要があります。注これはPOST 要求の必須パラメーターです。 |
| » description | string | false | none | なし |

次のページに続く

表 287 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » enabled | boolean | false | none | このアラートが有効/無効かどうかを判断します。注 このプロパティは、POST 要求の必須パラメーターです。 |
| » history_retention | integer | false | none | このアラートの履歴を保持する日数を指定します。指定できる値は 1〜99999 です。 注このプロパティは、POST 要求の必須パラメーターです。 |
| » frequency_min | integer | false | none | PEM データベースサーバーでアラートを確認する頻度を分単位で定義します。指定できる値は 1〜65534 です。 注このプロパティは、POST 要求の必須パラメーターです。 |
| » auto_created | boolean | false | none | アラートを自動作成するかどうかを指定します。 |
| » template_id | string | false | none | アラートテンプレートを定義します。 |
| » params | *Platform Compatibility* ] | false | none | アラートを確認するためにアラートテンプレートが使用するパラメーター/値のペアのリスト。注 アラートテンプレートには、all パラメーターを定義する必要があります。 |
| » paramname | string | false | none | パラメーターの名前。注このプロパティは、アラートテンプレートパラメーターリストに存在する必要があります。例 |
| » paramvalue | string | false | none | 指定されたパラメーターの値 |
| » param_names | [string] | false | none | パラメーターの名前を定義します。 |
| » param_types | string | false | none | パラメーターのタイプを定義します。例 INTEGER/STRING.. |

次のページに続く

表 287 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » params_units | [string] | false | none | パラメーターに使用される単位を定義します。例 'DAYS'、'True or False'、'Hours'、'Minutes' など |
| » threshold_unit | string | false | none | MB/kB/秒/日/... などの、しきい値の計算に使用される単位 この値は、GUI でのみ使用されます。 |
| » operator | string | false | none | しきい値をチェックするための演算子を定義します。有効な文字列は>"および"<"です。注 このプロパティは、POST 要求の必須パラメーターです。 |
| » low_threshold_value | number | false | none | 低アラートをトリガーするためのしきい値。このプロパティには、有効な整数/浮動小数点値を指定できます。注これは POST 要求の必須パラメーターです。 |
| » medium_threshold_value | number | false | none | 中アラートをトリガーするためのしきい値。このプロパティには、有効な整数/浮動小数点値を指定できます。注これは POST 要求の必須パラメーターです。 |
| » high_threshold_value | number | false | none | 高度なアラートをトリガーするためのしきい値。任意の有効な整数/浮動小数点値を指定できます。注これは POST 要求の必須パラメーターです。 |
| » send_email | boolean | false | none | アラートの状態が変更されたときに電子メールを送信するかどうかを決定します。 |

次のページに続く

表 287 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » all_alert_enable | boolean | false | none | アラートの状態が変更されたときに、パラメーター 'email_group_id' で指定されたグループにメール通知を送信するかどうかを決定します。 |
| » email_group_id | integer | false | none | アラートの状態が変更されたときに電子メール通知の送信先の電子メールグループを定義します。 |
| » low_alert_enable | boolean | false | none | アラートの状態が LOW に設定されているときに、パラメーター 'low_email_group_id' で指定されたグループにメール通知を送信するかどうかを決定します。 |
| » low_email_group_id | integer | false | none | アラートの状態が LOW に変更されたときに電子メール通知の送信先の電子メールグループを定義します。 |
| » med_alert_enable | boolean | false | none | アラートの状態が MEDIUM に設定されている場合、パラメーター 'medium_email_group_id' で指定されたグループにメール通知を送信するかどうかを決定します。 |
| » med_email_group_id | integer | false | none | アラートの状態が MEDIUM に変更されたときにメール通知の送信先のメールグループを定義します。 |
| » high_alert_enable | boolean | false | none | アラートの状態が HIGH に設定されているときに、パラメーター 'high_email_group_id' で指定されたグループに電子メール通知を送信するかどうかを決定します。 |

次のページに続く

表 287 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » high_email_group_id | integer | false | none | アラートの状態が HIGH に変更されたときに電子メール通知の送信先の電子メールグループを定義します。 |
| » send_trap | boolean | false | none | アラートの状態が変更されたときに SNMP トラップ通知を送信するかどうかを決定します。 |
| » snmp_trap_version | integer | false | none | SNMP トラップバージョンを定義します。サポートされている SNMP トラップバージョンは 1、2、または 3 です |
| » low_send_trap | boolean | false | none | アラートの状態が LOW に変更されたときに SNMP トラップ通知を送信するかどうかを決定します。 |
| » med_send_trap | boolean | false | none | アラートの状態が MEDIUM に変更されたときに SNMP トラップ通知を送信するかどうかを決定します。 |
| » high_send_trap | boolean | false | none | アラートの状態が HIGH に変わったときに SNMP トラップ通知を送信するかどうかを決定します。 |
| » cleared_alert_enable | boolean | false | none | ' クリアアラート' の通知をミュートするかどうかを決定します。 |
| » execute_script | boolean | false | none | アラートの状態が LOW/MEDIUM/HIGH に変更された場合に、提供されたスクリプトを実行するかどうかを決定します。 |
| » execute_script_on_clear | boolean | false | none | アラートの状態がクリアされた場合に、提供されたスクリプトを実行するかどうかを決定します。 |

次のページに続く

表 287 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » execute_script_on_pem_serve | boolean | false | none | 提供されたスクリプトをPEM ホストまたはエージェントがオブジェクトを監視しているホストで実行するかどうかを決定します。 |
| » script_code | string | false | none | 実行するシェルまたはバッチスクリプト。詳細については、ドキュメントをご覧ください。 |
| » submit_to_nagios | boolean | false | none | アラートの状態が変更されたときに nagios に通知を送信するかどうかを決定します。 |
| » override_default_config | boolean | false | none | デフォルトの Webhook 構成をオーバーライドするかどうかを決定します。 |
| » send_notification | boolean | false | none | Webhook 送信通知を有効にするかどうかを決定します。 |
| » low_webhook_ids | [integer] | false | none | 低アラートが構成されている Webhook ID |
| » med_webhook_ids | [integer] | false | none | 中アラートが構成されている Webhook ID |
| » high_webhook_ids | [integer] | false | none | 高アラートが構成されている Webhook ID |
| » cleared_webhook_ids | [integer] | false | none | クリアされたアラートが構成されている Webhook ID |

### 列挙値

| Property | Value |
|---|---|
| paramname | Exclude tables smaller than |
| paramname | mount point |
| paramname | Host IP Address |
| paramname | Time interval to monitor |
| paramname | Time since idle |
| paramname | Per CPU utilization (K%) |

> **注釈**
>
> この操作には認証が必要ありません

### post_*alert_config_server*{server_id}*database*{database_name}

#### コードサンプル

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/
↪database/{database_name} \
  -H Content-Type: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Content-Type: application/json,
  X-Auth-Token: string
}

r = requests.post(https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/
↪database/{database_name}, headers = headers)

print(r.json())
```

```
const inputBody = {
  "alert_name": "string",
  "alert_template": 0,
  "description": "string",
  "enabled": true,
  "history_retention": 0,
  "frequency_min": 0,
  "operator": "string",
  "low_threshold_value": 0,
  "medium_threshold_value": 0,
  "high_threshold_value": 0,
  "all_alert_enable": true,
  "email_group_id": 1,
  "low_alert_enable": true,
  "low_email_group_id": 1,
  "med_alert_enable": true,
  "med_email_group_id": 1,
  "high_alert_enable": true,
  "high_email_group_id": 1,
```

(次のページに続く)

(前のページからの続き)

```
  "send_trap": false,
  "snmp_trap_version": 0,
  "low_send_trap": true,
  "med_send_trap": true,
  "high_send_trap": true,
  "cleared_alert_enable": true,
  "params": [],
  "execute_script": true,
  "execute_script_on_clear": true,
  "execute_script_on_pem_server": true,
  "script_code": "string",
  "submit_to_nagios": true,
  "override_default_config": true,
  "send_notification": false,
  "low_webhook_ids": [],
  "med_webhook_ids": [],
  "high_webhook_ids": [],
  "cleared_webhook_ids": []
};
const headers = {
  Content-Type:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/database/
↪{database_name},
{
  method: POST,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

POST /alert/config/server/{server_id}/database/{database_name}

データベースレベルの Alert を作成します。

**ボディパラメーター**

```
{
  "alert_name": "string",
  "alert_template": 0,
```

(次のページに続く)

(前のページからの続き)


```
  "description": "string",
  "enabled": true,
  "history_retention": 0,
  "frequency_min": 0,
  "operator": "string",
  "low_threshold_value": 0,
  "medium_threshold_value": 0,
  "high_threshold_value": 0,
  "all_alert_enable": true,
  "email_group_id": 1,
  "low_alert_enable": true,
  "low_email_group_id": 1,
  "med_alert_enable": true,
  "med_email_group_id": 1,
  "high_alert_enable": true,
  "high_email_group_id": 1,
  "send_trap": false,
  "snmp_trap_version": 0,
  "low_send_trap": true,
  "med_send_trap": true,
  "high_send_trap": true,
  "cleared_alert_enable": true,
  "params": [],
  "execute_script": true,
  "execute_script_on_clear": true,
  "execute_script_on_pem_server": true,
  "script_code": "string",
  "submit_to_nagios": true,
  "override_default_config": true,
  "send_notification": false,
  "low_webhook_ids": [],
  "med_webhook_ids": [],
  "high_webhook_ids": [],
  "cleared_webhook_ids": []
}
```


### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| database_name | path | string | true | データベース名 |
| X-Auth-Token | header | string | true | 承認トークン |

次のページに続く

表 289 – 前のページからの続き

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| body | body | *AlertPayload* | true | なし |

### 応答例

#### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ステータス 200 | None |

#### レスポンススキーマ

**注釈**

この操作には認証が必要ありません

### get_*alert_config_server*{server_id}*database*{database_name}_{alert_id}

#### コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/
↪database/{database_name}/{alert_id} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/
↪database/{database_name}/{alert_id}, headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch(
  "https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/database/
↪{database_name}/{alert_id}",
  {
    method: "GET",

    headers: headers,
  },
)
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

`GET /alert/config/server/{server_id}/database/{database_name}/{alert_id}`

server_id、database_name、および alert_id ごとにデータベースレベルの `Alerts` を取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| database_name | path | string | true | データベース名 |
| alert_id | path | integer | true | アラート ID |
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例

```
{
  "id": 0,
  "alert_name": "string",
  "alert_template": 0,
  "description": "string",
  "enabled": true,
  "history_retention": 0,
  "frequency_min": 0,
  "auto_created": true,
  "template_id": "string",
```

(次のページに続く)

(前のページからの続き)

```
  "params": [],
  "param_names": ["string"],
  "param_types": "string",
  "params_units": ["string"],
  "threshold_unit": "string",
  "operator": "string",
  "low_threshold_value": 0,
  "medium_threshold_value": 0,
  "high_threshold_value": 0,
  "send_email": true,
  "all_alert_enable": true,
  "email_group_id": 0,
  "low_alert_enable": true,
  "low_email_group_id": 0,
  "med_alert_enable": true,
  "med_email_group_id": 0,
  "high_alert_enable": true,
  "high_email_group_id": 0,
  "send_trap": true,
  "snmp_trap_version": 0,
  "low_send_trap": true,
  "med_send_trap": true,
  "high_send_trap": true,
  "cleared_alert_enable": true,
  "execute_script": true,
  "execute_script_on_clear": true,
  "execute_script_on_pem_server": true,
  "script_code": "string",
  "submit_to_nagios": true,
  "override_default_config": true,
  "send_notification": true,
  "low_webhook_ids": [0],
  "med_webhook_ids": [0],
  "high_webhook_ids": [0],
  "cleared_webhook_ids": [0]
}
```

応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | アラートスレッドについて |

**注釈**

この操作には認証が必要ありません

### put_*alert_config_server*{server_id}*database*{database_name}_{alert_id}

コードサンプル

```
#  You can also use wget

curl -X PUT https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/
↪database/{database_name}/{alert_id} \
  -H Content-Type: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Content-Type: application/json,
  X-Auth-Token: string
}

r = requests.put(https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/
↪database/{database_name}/{alert_id}, headers = headers)

print(r.json())
```

```
const inputBody = {
  "alert_name": "string",
  "alert_template": 0,
  "description": "string",
  "enabled": true,
  "history_retention": 0,
  "frequency_min": 0,
  "operator": "string",
  "low_threshold_value": 0,
  "medium_threshold_value": 0,
  "high_threshold_value": 0,
  "all_alert_enable": true,
```

(次のページに続く)

(前のページからの続き)

```
  "email_group_id": 1,
  "low_alert_enable": true,
  "low_email_group_id": 1,
  "med_alert_enable": true,
  "med_email_group_id": 1,
  "high_alert_enable": true,
  "high_email_group_id": 1,
  "send_trap": false,
  "snmp_trap_version": 0,
  "low_send_trap": true,
  "med_send_trap": true,
  "high_send_trap": true,
  "cleared_alert_enable": true,
  "params": [],
  "execute_script": true,
  "execute_script_on_clear": true,
  "execute_script_on_pem_server": true,
  "script_code": "string",
  "submit_to_nagios": true,
  "override_default_config": true,
  "send_notification": false,
  "low_webhook_ids": [],
  "med_webhook_ids": [],
  "high_webhook_ids": [],
  "cleared_webhook_ids": []
};
const headers = {
  Content-Type:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/database/
↪{database_name}/{alert_id},
{
  method: PUT,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

```
PUT /alert/config/server/{server_id}/database/{database_name}/{alert_id}
```

データベースレベルの Alert を更新します。

**ボディパラメーター**

```
{
  "alert_name": "string",
  "alert_template": 0,
  "description": "string",
  "enabled": true,
  "history_retention": 0,
  "frequency_min": 0,
  "operator": "string",
  "low_threshold_value": 0,
  "medium_threshold_value": 0,
  "high_threshold_value": 0,
  "all_alert_enable": true,
  "email_group_id": 1,
  "low_alert_enable": true,
  "low_email_group_id": 1,
  "med_alert_enable": true,
  "med_email_group_id": 1,
  "high_alert_enable": true,
  "high_email_group_id": 1,
  "send_trap": false,
  "snmp_trap_version": 0,
  "low_send_trap": true,
  "med_send_trap": true,
  "high_send_trap": true,
  "cleared_alert_enable": true,
  "params": [],
  "execute_script": true,
  "execute_script_on_clear": true,
  "execute_script_on_pem_server": true,
  "script_code": "string",
  "submit_to_nagios": true,
  "override_default_config": true,
  "send_notification": false,
  "low_webhook_ids": [],
  "med_webhook_ids": [],
  "high_webhook_ids": [],
  "cleared_webhook_ids": []
}
```

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| database_name | path | string | true | データベース名 |
| alert_id | path | integer | true | アラート ID |
| X-Auth-Token | header | string | true | 承認トークン |
| body | body | *AlertPayload* | true | なし |

### 応答例

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ステータス 200 | None |

### レスポンススキーマ

**注釈**

この操作には認証が必要ありません

## delete_*alert_config_server*{server_id}*database*{database_name}_{alert_id}

### コードサンプル

```
#  You can also use wget

curl -X DELETE https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/
↪database/{database_name}/{alert_id} \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  X-Auth-Token: string
}

r = requests.delete(https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/
↪database/{database_name}/{alert_id}, headers = headers)
```

(次のページに続く)

(前のページからの続き)

```
print(r.json())
```

```
const headers = {
  "X-Auth-Token": "string",
};

fetch(
  "https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/database/
↪{database_name}/{alert_id}",
  {
    method: "DELETE",

    headers: headers,
  },
)
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

`DELETE /alert/config/server/{server_id}/database/{database_name}/{alert_id}`

`Alert` オブジェクトを削除します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| database_name | path | string | true | データベース名 |
| alert_id | path | integer | true | アラート ID |
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例

#### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 削除が成功しました。 | None |

レスポンススキーマ

> **注釈**
>
> この操作には認証が必要ありません

### get_*alert_config_server*{server_id}*database*{database_name}*schema*{schema_name}

コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}, headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch(
  "https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/database/
↪{database_name}/schema/{schema_name}",
  {
    method: "GET",
```

(次のページに続く)

(前のページからの続き)

```
    headers: headers,
  },
)
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /alert/config/server/{server_id}/database/{database_name}/schema/{schema_name}

server_id、database_name、および schema_name によってスキーマレベルの Alerts を取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| database_name | path | string | true | データベース名 |
| schema_name | path | string | true | スキーマ名 |
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例

```
[
  {
    "id": 0,
    "alert_name": "string",
    "alert_template": 0,
    "description": "string",
    "enabled": true,
    "history_retention": 0,
    "frequency_min": 0,
    "auto_created": true,
    "template_id": "string",
    "params": [],
    "param_names": ["string"],
    "param_types": "string",
    "params_units": ["string"],
    "threshold_unit": "string",
    "operator": "string",
    "low_threshold_value": 0,
    "medium_threshold_value": 0,
```

(次のページに続く)

(前のページからの続き)


```
    "high_threshold_value": 0,
    "send_email": true,
    "all_alert_enable": true,
    "email_group_id": 0,
    "low_alert_enable": true,
    "low_email_group_id": 0,
    "med_alert_enable": true,
    "med_email_group_id": 0,
    "high_alert_enable": true,
    "high_email_group_id": 0,
    "send_trap": true,
    "snmp_trap_version": 0,
    "low_send_trap": true,
    "med_send_trap": true,
    "high_send_trap": true,
    "cleared_alert_enable": true,
    "execute_script": true,
    "execute_script_on_clear": true,
    "execute_script_on_pem_server": true,
    "script_code": "string",
    "submit_to_nagios": true,
    "override_default_config": true,
    "send_notification": true,
    "low_webhook_ids": [0],
    "med_webhook_ids": [0],
    "high_webhook_ids": [0],
    "cleared_webhook_ids": [0]
  }
]
```


### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | Inline |

### レスポンススキーマ

ステータスコード **200**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| _anonymous_ | *Platform Compatibility* ] | false | none | なし |
| » id | integer | false | none | アラート ID。注このプロパティは、POST 要求には適用されません。 |
| » alert_name | string | false | none | アラートの名前。注 このプロパティは、POST 要求の必須パラメーターです。 |
| » alert_template | integer | false | none | アラートテンプレートの ID。このプロパティは、正の整数値である必要があります。注これはPOST 要求の必須パラメーターです。 |
| » description | string | false | none | なし |
| » enabled | boolean | false | none | このアラートが有効/無効かどうかを判断します。注 このプロパティは、POST 要求の必須パラメーターです。 |
| » history_retention | integer | false | none | このアラートの履歴を保持する日数を指定します。指定できる値は 1 〜99999 です。 注このプロパティは、POST 要求の必須パラメーターです。 |
| » frequency_min | integer | false | none | PEM データベースサーバーでアラートを確認する頻度を分単位で定義します。指定できる値は 1 〜65534 です。 注このプロパティは、POST 要求の必須パラメーターです。 |
| » auto_created | boolean | false | none | アラートを自動作成するかどうかを指定します。 |
| » template_id | string | false | none | アラートテンプレートを定義します。 |

次のページに続く

表 299 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » params | *Platform Compatibility* ] | false | none | アラートを確認するためにアラートテンプレートが使用するパラメーター/値のペアのリスト。注アラートテンプレートには、all パラメーターを定義する必要があります。 |
| » paramname | string | false | none | パラメーターの名前。注このプロパティは、アラートテンプレートパラメーターリストに存在する必要があります。例 |
| » paramvalue | string | false | none | 指定されたパラメーターの値 |
| » param_names | [string] | false | none | パラメーターの名前を定義します。 |
| » param_types | string | false | none | パラメーターのタイプを定義します。例 INTEGER/STRING.. |
| » params_units | [string] | false | none | パラメーターに使用される単位を定義します。例 'DAYS'、'True or False'、'Hours'、'Minutes' など |
| » threshold_unit | string | false | none | MB/kB/秒/日/... などの、しきい値の計算に使用される単位 この値は、GUI でのみ使用されます。 |
| » operator | string | false | none | しきい値をチェックするための演算子を定義します。有効な文字列は>"および"<"です。注このプロパティは、POST 要求の必須パラメーターです。 |
| » low_threshold_value | number | false | none | 低アラートをトリガーするためのしきい値。このプロパティには、有効な整数/浮動小数点値を指定できます。注これは POST 要求の必須パラメーターです。 |

次のページに続く

表 299 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » medium_threshold_value | number | false | none | 中アラートをトリガーするためのしきい値。このプロパティには、有効な整数/浮動小数点値を指定できます。注これはPOST要求の必須パラメーターです。 |
| » high_threshold_value | number | false | none | 高度なアラートをトリガーするためのしきい値。任意の有効な整数/浮動小数点値を指定できます。注これはPOST要求の必須パラメーターです。 |
| » send_email | boolean | false | none | アラートの状態が変更されたときに電子メールを送信するかどうかを決定します。 |
| » all_alert_enable | boolean | false | none | アラートの状態が変更されたときに、パラメーター 'email_group_id' で指定されたグループにメール通知を送信するかどうかを決定します。 |
| » email_group_id | integer | false | none | アラートの状態が変更されたときに電子メール通知の送信先の電子メールグループを定義します。 |
| » low_alert_enable | boolean | false | none | アラートの状態がLOWに設定されているときに、パラメーター 'low_email_group_id' で指定されたグループにメール通知を送信するかどうかを決定します。 |
| » low_email_group_id | integer | false | none | アラートの状態がLOWに変更されたときに電子メール通知の送信先の電子メールグループを定義します。 |

次のページに続く

表 299 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » med_alert_enable | boolean | false | none | アラートの状態が MEDIUM に設定されている場合、パラメーター 'medium_email_group_id' で指定されたグループにメール通知を送信するかどうかを決定します。 |
| » med_email_group_id | integer | false | none | アラートの状態が MEDIUM に変更されたときにメール通知の送信先のメールグループを定義します。 |
| » high_alert_enable | boolean | false | none | アラートの状態が HIGH に設定されているときに、パラメーター 'high_email_group_id' で指定されたグループに電子メール通知を送信するかどうかを決定します。 |
| » high_email_group_id | integer | false | none | アラートの状態が HIGH に変更されたときに電子メール通知の送信先の電子メールグループを定義します。 |
| » send_trap | boolean | false | none | アラートの状態が変更されたときに SNMP トラップ通知を送信するかどうかを決定します。 |
| » snmp_trap_version | integer | false | none | SNMP トラップバージョンを定義します。サポートされている SNMP トラップバージョンは 1、2、または 3 です |
| » low_send_trap | boolean | false | none | アラートの状態が LOW に変更されたときに SNMP トラップ通知を送信するかどうかを決定します。 |

次のページに続く

表 299 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » med_send_trap | boolean | false | none | アラートの状態が MEDIUM に変更されたときに SNMP トラップ通知を送信するかどうかを決定します。 |
| » high_send_trap | boolean | false | none | アラートの状態が HIGH に変わったときに SNMP トラップ通知を送信するかどうかを決定します。 |
| » cleared_alert_enable | boolean | false | none | ' クリアアラート' の通知をミュートするかどうかを決定します。 |
| » execute_script | boolean | false | none | アラートの状態が LOW/MEDIUM/HIGH に変更された場合に、提供されたスクリプトを実行するかどうかを決定します。 |
| » execute_script_on_clear | boolean | false | none | アラートの状態がクリアされた場合に、提供されたスクリプトを実行するかどうかを決定します。 |
| » execute_script_on_pem_serve | boolean | false | none | 提供されたスクリプトを PEM ホストまたはエージェントがオブジェクトを監視しているホストで実行するかどうかを決定します。 |
| » script_code | string | false | none | 実行するシェルまたはバッチスクリプト。詳細については、ドキュメントをご覧ください。 |
| » submit_to_nagios | boolean | false | none | アラートの状態が変更されたときに nagios に通知を送信するかどうかを決定します。 |
| » override_default_config | boolean | false | none | デフォルトの Webhook 構成をオーバーライドするかどうかを決定します。 |

次のページに続く

表 299 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » send_notification | boolean | false | none | Webhook 送信通知を有効にするかどうかを決定します。 |
| » low_webhook_ids | [integer] | false | none | 低アラートが構成されている Webhook ID |
| » med_webhook_ids | [integer] | false | none | 中アラートが構成されている Webhook ID |
| » high_webhook_ids | [integer] | false | none | 高アラートが構成されている Webhook ID |
| » cleared_webhook_ids | [integer] | false | none | クリアされたアラートが構成されている Webhook ID |

#### 列挙値

| Property | Value |
|---|---|
| paramname | Exclude tables smaller than |
| paramname | mount point |
| paramname | Host IP Address |
| paramname | Time interval to monitor |
| paramname | Time since idle |
| paramname | Per CPU utilization (K%) |

**注釈**

この操作には認証が必要ありません

### post_*alert_config_server*{server_id}*database*{database_name}*schema*{schema_name}

#### コードサンプル

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name} \
  -H Content-Type: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Content-Type: application/json,
```

(次のページに続く)

(前のページからの続き)

```
  X-Auth-Token: string
}

r = requests.post(https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}, headers = headers)

print(r.json())
```

```
const inputBody = {
  "alert_name": "string",
  "alert_template": 0,
  "description": "string",
  "enabled": true,
  "history_retention": 0,
  "frequency_min": 0,
  "operator": "string",
  "low_threshold_value": 0,
  "medium_threshold_value": 0,
  "high_threshold_value": 0,
  "all_alert_enable": true,
  "email_group_id": 1,
  "low_alert_enable": true,
  "low_email_group_id": 1,
  "med_alert_enable": true,
  "med_email_group_id": 1,
  "high_alert_enable": true,
  "high_email_group_id": 1,
  "send_trap": false,
  "snmp_trap_version": 0,
  "low_send_trap": true,
  "med_send_trap": true,
  "high_send_trap": true,
  "cleared_alert_enable": true,
  "params": [],
  "execute_script": true,
  "execute_script_on_clear": true,
  "execute_script_on_pem_server": true,
  "script_code": "string",
  "submit_to_nagios": true,
  "override_default_config": true,
  "send_notification": false,
  "low_webhook_ids": [],
  "med_webhook_ids": [],
  "high_webhook_ids": [],
```

(次のページに続く)

(前のページからの続き)

```
  "cleared_webhook_ids": []
};
const headers = {
  Content-Type:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/database/
↪{database_name}/schema/{schema_name},
{
  method: POST,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

POST /alert/config/server/{server_id}/database/{database_name}/schema/{schema_name}

スキーマレベルの Alert を作成します。

**ボディパラメーター**

```
{
  "alert_name": "string",
  "alert_template": 0,
  "description": "string",
  "enabled": true,
  "history_retention": 0,
  "frequency_min": 0,
  "operator": "string",
  "low_threshold_value": 0,
  "medium_threshold_value": 0,
  "high_threshold_value": 0,
  "all_alert_enable": true,
  "email_group_id": 1,
  "low_alert_enable": true,
  "low_email_group_id": 1,
  "med_alert_enable": true,
  "med_email_group_id": 1,
  "high_alert_enable": true,
  "high_email_group_id": 1,
  "send_trap": false,
```

(次のページに続く)

(前のページからの続き)


```
  "snmp_trap_version": 0,
  "low_send_trap": true,
  "med_send_trap": true,
  "high_send_trap": true,
  "cleared_alert_enable": true,
  "params": [],
  "execute_script": true,
  "execute_script_on_clear": true,
  "execute_script_on_pem_server": true,
  "script_code": "string",
  "submit_to_nagios": true,
  "override_default_config": true,
  "send_notification": false,
  "low_webhook_ids": [],
  "med_webhook_ids": [],
  "high_webhook_ids": [],
  "cleared_webhook_ids": []
}
```


### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| database_name | path | string | true | データベース名 |
| schema_name | path | string | true | スキーマ名 |
| X-Auth-Token | header | string | true | 承認トークン |
| body | body | *AlertPayload* | true | なし |

### 応答例

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ステータス 200 | None |

### レスポンススキーマ

> **注釈**
>
> この操作には認証が必要ありません

## get_*alert_config_server*{server_id}*database*{database_name}*schema*{schema_name}_{alert_id}

### コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/{alert_id} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/{alert_id}, headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch(
  "https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/database/
↪{database_name}/schema/{schema_name}/{alert_id}",
  {
    method: "GET",

    headers: headers,
  },
)
  .then(function (res) {
```

(次のページに続く)

(前のページからの続き)

```
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /alert/config/server/{server_id}/database/{database_name}/schema/{schema_name}/{alert_id}

server_id、database_name、schema_name、および alert_id ごとにスキーマレベルの Alerts を取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| database_name | path | string | true | データベース名 |
| schema_name | path | string | true | スキーマ名 |
| alert_id | path | integer | true | アラート ID |
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例

```
{
  "id": 0,
  "alert_name": "string",
  "alert_template": 0,
  "description": "string",
  "enabled": true,
  "history_retention": 0,
  "frequency_min": 0,
  "auto_created": true,
  "template_id": "string",
  "params": [],
  "param_names": ["string"],
  "param_types": "string",
  "params_units": ["string"],
  "threshold_unit": "string",
  "operator": "string",
  "low_threshold_value": 0,
  "medium_threshold_value": 0,
  "high_threshold_value": 0,
  "send_email": true,
  "all_alert_enable": true,
  "email_group_id": 0,
```

(次のページに続く)

(前のページからの続き)

```
  "low_alert_enable": true,
  "low_email_group_id": 0,
  "med_alert_enable": true,
  "med_email_group_id": 0,
  "high_alert_enable": true,
  "high_email_group_id": 0,
  "send_trap": true,
  "snmp_trap_version": 0,
  "low_send_trap": true,
  "med_send_trap": true,
  "high_send_trap": true,
  "cleared_alert_enable": true,
  "execute_script": true,
  "execute_script_on_clear": true,
  "execute_script_on_pem_server": true,
  "script_code": "string",
  "submit_to_nagios": true,
  "override_default_config": true,
  "send_notification": true,
  "low_webhook_ids": [0],
  "med_webhook_ids": [0],
  "high_webhook_ids": [0],
  "cleared_webhook_ids": [0]
}
```

応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | アラートスレッドについて |

**注釈**

この操作には認証が必要ありません

**put_*alert_config_server*{server_id}*database*{database_name}*schema*{schema_name}_{alert_id}**

コードサンプル

```
#  You can also use wget
```

(次のページに続く)

(前のページからの続き)

```
curl -X PUT https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/{alert_id} \
  -H Content-Type: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Content-Type: application/json,
  X-Auth-Token: string
}

r = requests.put(https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/{alert_id}, headers = headers)

print(r.json())
```

```
const inputBody = {
  "alert_name": "string",
  "alert_template": 0,
  "description": "string",
  "enabled": true,
  "history_retention": 0,
  "frequency_min": 0,
  "operator": "string",
  "low_threshold_value": 0,
  "medium_threshold_value": 0,
  "high_threshold_value": 0,
  "all_alert_enable": true,
  "email_group_id": 1,
  "low_alert_enable": true,
  "low_email_group_id": 1,
  "med_alert_enable": true,
  "med_email_group_id": 1,
  "high_alert_enable": true,
  "high_email_group_id": 1,
  "send_trap": false,
  "snmp_trap_version": 0,
  "low_send_trap": true,
  "med_send_trap": true,
  "high_send_trap": true,
  "cleared_alert_enable": true,
  "params": [],
  "execute_script": true,
  "execute_script_on_clear": true,
```

(次のページに続く)

(前のページからの続き)

```
  "execute_script_on_pem_server": true,
  "script_code": "string",
  "submit_to_nagios": true,
  "override_default_config": true,
  "send_notification": false,
  "low_webhook_ids": [],
  "med_webhook_ids": [],
  "high_webhook_ids": [],
  "cleared_webhook_ids": []
};
const headers = {
  Content-Type:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/database/
↪{database_name}/schema/{schema_name}/{alert_id},
{
  method: PUT,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

PUT /alert/config/server/{server_id}/database/{database_name}/schema/{schema_name}/{alert_id}

スキーマレベルの Alert を更新します。

**ボディパラメーター**

```
{
  "alert_name": "string",
  "alert_template": 0,
  "description": "string",
  "enabled": true,
  "history_retention": 0,
  "frequency_min": 0,
  "operator": "string",
  "low_threshold_value": 0,
  "medium_threshold_value": 0,
  "high_threshold_value": 0,
```

(次のページに続く)

(前のページからの続き)


```
  "all_alert_enable": true,
  "email_group_id": 1,
  "low_alert_enable": true,
  "low_email_group_id": 1,
  "med_alert_enable": true,
  "med_email_group_id": 1,
  "high_alert_enable": true,
  "high_email_group_id": 1,
  "send_trap": false,
  "snmp_trap_version": 0,
  "low_send_trap": true,
  "med_send_trap": true,
  "high_send_trap": true,
  "cleared_alert_enable": true,
  "params": [],
  "execute_script": true,
  "execute_script_on_clear": true,
  "execute_script_on_pem_server": true,
  "script_code": "string",
  "submit_to_nagios": true,
  "override_default_config": true,
  "send_notification": false,
  "low_webhook_ids": [],
  "med_webhook_ids": [],
  "high_webhook_ids": [],
  "cleared_webhook_ids": []
}
```


### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| database_name | path | string | true | データベース名 |
| schema_name | path | string | true | スキーマ名 |
| alert_id | path | integer | true | アラート ID |
| X-Auth-Token | header | string | true | 承認トークン |
| body | body | *AlertPayload* | true | なし |

### 応答例

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ステータス 200 | None |

### レスポンススキーマ

> **注釈**
>
> この操作には認証が必要ありません

## delete_*alert_config_server*{server_id}*database*{database_name}*schema*{schema_name}_{alert_id}

### コードサンプル

```
#  You can also use wget

curl -X DELETE https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/{alert_id} \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  X-Auth-Token: string
}

r = requests.delete(https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/{alert_id}, headers = headers)

print(r.json())
```

```
const headers = {
  "X-Auth-Token": "string",
};

fetch(
  "https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/database/
↪{database_name}/schema/{schema_name}/{alert_id}",
  {
    method: "DELETE",
```

(次のページに続く)

(前のページからの続き)

```
    headers: headers,
  },
)
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

`DELETE /alert/config/server/{server_id}/database/{database_name}/schema/{schema_name}/{alert_id}`

`Alert` オブジェクトを削除します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| database_name | path | string | true | データベース名 |
| schema_name | path | string | true | スキーマ名 |
| alert_id | path | integer | true | アラート ID |
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 削除が成功しました。 | None |

### レスポンススキーマ

**注釈**

この操作には認証が必要ありません

## get_*alert_config_server*{server_id}*database*{database_name}*schema*{schema_name}*table*{table_name}

### コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/table/{table_name} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/table/{table_name}, headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch(
  "https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/database/
↪{database_name}/schema/{schema_name}/table/{table_name}",
  {
    method: "GET",

    headers: headers,
  },
)
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /alert/config/server/{server_id}/database/{database_name}/schema/{schema_name}/table/{table_name}

server_id、database_name、schema_name、および table_name ごとにテーブルレベルの Alerts を取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| database_name | path | string | true | データベース名 |
| schema_name | path | string | true | スキーマ名 |
| table_name | path | string | true | テーブル名 |
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例


```
[
  {
    "id": 0,
    "alert_name": "string",
    "alert_template": 0,
    "description": "string",
    "enabled": true,
    "history_retention": 0,
    "frequency_min": 0,
    "auto_created": true,
    "template_id": "string",
    "params": [],
    "param_names": ["string"],
    "param_types": "string",
    "params_units": ["string"],
    "threshold_unit": "string",
    "operator": "string",
    "low_threshold_value": 0,
    "medium_threshold_value": 0,
    "high_threshold_value": 0,
    "send_email": true,
    "all_alert_enable": true,
    "email_group_id": 0,
    "low_alert_enable": true,
    "low_email_group_id": 0,
    "med_alert_enable": true,
    "med_email_group_id": 0,
    "high_alert_enable": true,
    "high_email_group_id": 0,
    "send_trap": true,
    "snmp_trap_version": 0,
    "low_send_trap": true,
    "med_send_trap": true,
    "high_send_trap": true,
```


(次のページに続く)

(前のページからの続き)


```
    "cleared_alert_enable": true,
    "execute_script": true,
    "execute_script_on_clear": true,
    "execute_script_on_pem_server": true,
    "script_code": "string",
    "submit_to_nagios": true,
    "override_default_config": true,
    "send_notification": true,
    "low_webhook_ids": [0],
    "med_webhook_ids": [0],
    "high_webhook_ids": [0],
    "cleared_webhook_ids": [0]
  }
]
```


### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | Inline |

### レスポンススキーマ

ステータスコード **200**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| _anonymous_ | *Platform Compatibility* ] | false | none | なし |
| » id | integer | false | none | アラート ID。注このプロパティは、POST 要求には適用されません。 |
| » alert_name | string | false | none | アラートの名前。注 このプロパティは、POST 要求の必須パラメーターです。 |

次のページに続く

表 311 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » alert_template | integer | false | none | アラートテンプレートの ID。このプロパティは、正の整数値である必要があります。注これはPOST 要求の必須パラメーターです。 |
| » description | string | false | none | なし |
| » enabled | boolean | false | none | このアラートが有効/無効かどうかを判断します。注 このプロパティは、POST 要求の必須パラメーターです。 |
| » history_retention | integer | false | none | このアラートの履歴を保持する日数を指定します。指定できる値は 1 〜99999 です。 注このプロパティは、POST 要求の必須パラメーターです。 |
| » frequency_min | integer | false | none | PEM データベースサーバーでアラートを確認する頻度を分単位で定義します。指定できる値は 1 〜65534 です。 注このプロパティは、POST 要求の必須パラメーターです。 |
| » auto_created | boolean | false | none | アラートを自動作成するかどうかを指定します。 |
| » template_id | string | false | none | アラートテンプレートを定義します。 |
| » params | *Platform Compatibility* ] | false | none | アラートを確認するためにアラートテンプレートが使用するパラメーター/値のペアのリスト。注 アラートテンプレートには、all パラメーターを定義する必要があります。 |
| » paramname | string | false | none | パラメーターの名前。注このプロパティは、アラートテンプレートパラメーターリストに存在する必要があります。例 |

次のページに続く

表 311 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » paramvalue | string | false | none | 指定されたパラメーターの値 |
| » param_names | [string] | false | none | パラメーターの名前を定義します。 |
| » param_types | string | false | none | パラメーターのタイプを定義します。例 INTEGER/STRING.. |
| » params_units | [string] | false | none | パラメーターに使用される単位を定義します。例 'DAYS'、'True or False'、'Hours'、'Minutes' など |
| » threshold_unit | string | false | none | MB/kB/秒/日/... などの、しきい値の計算に使用される単位 この値は、GUI でのみ使用されます。 |
| » operator | string | false | none | しきい値をチェックするための演算子を定義します。有効な文字列は>"および"<"です。注 このプロパティは、POST 要求の必須パラメーターです。 |
| » low_threshold_value | number | false | none | 低アラートをトリガーするためのしきい値。このプロパティには、有効な整数/浮動小数点値を指定できます。注これは POST 要求の必須パラメーターです。 |
| » medium_threshold_value | number | false | none | 中アラートをトリガーするためのしきい値。このプロパティには、有効な整数/浮動小数点値を指定できます。注これは POST 要求の必須パラメーターです。 |

次のページに続く

表 311 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » high_threshold_value | number | false | none | 高度なアラートをトリガーするためのしきい値。任意の有効な整数/浮動小数点値を指定できます。注これはPOST要求の必須パラメーターです。 |
| » send_email | boolean | false | none | アラートの状態が変更されたときに電子メールを送信するかどうかを決定します。 |
| » all_alert_enable | boolean | false | none | アラートの状態が変更されたときに、パラメーター 'email_group_id' で指定されたグループにメール通知を送信するかどうかを決定します。 |
| » email_group_id | integer | false | none | アラートの状態が変更されたときに電子メール通知の送信先の電子メールグループを定義します。 |
| » low_alert_enable | boolean | false | none | アラートの状態がLOWに設定されているときに、パラメーター 'low_email_group_id' で指定されたグループにメール通知を送信するかどうかを決定します。 |
| » low_email_group_id | integer | false | none | アラートの状態がLOWに変更されたときに電子メール通知の送信先の電子メールグループを定義します。 |
| » med_alert_enable | boolean | false | none | アラートの状態がMEDIUMに設定されている場合、パラメーター 'medium_email_group_id' で指定されたグループにメール通知を送信するかどうかを決定します。 |

次のページに続く

表 311 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » med_email_group_id | integer | false | none | アラートの状態が MEDIUM に変更されたときにメール通知の送信先のメールグループを定義します。 |
| » high_alert_enable | boolean | false | none | アラートの状態が HIGH に設定されているときに、パラメーター 'high_email_group_id' で指定されたグループに電子メール通知を送信するかどうかを決定します。 |
| » high_email_group_id | integer | false | none | アラートの状態が HIGH に変更されたときに電子メール通知の送信先の電子メールグループを定義します。 |
| » send_trap | boolean | false | none | アラートの状態が変更されたときに SNMP トラップ通知を送信するかどうかを決定します。 |
| » snmp_trap_version | integer | false | none | SNMP トラップバージョンを定義します。サポートされている SNMP トラップバージョンは 1、2、または 3 です |
| » low_send_trap | boolean | false | none | アラートの状態が LOW に変更されたときに SNMP トラップ通知を送信するかどうかを決定します。 |
| » med_send_trap | boolean | false | none | アラートの状態が MEDIUM に変更されたときに SNMP トラップ通知を送信するかどうかを決定します。 |
| » high_send_trap | boolean | false | none | アラートの状態が HIGH に変わったときに SNMP トラップ通知を送信するかどうかを決定します。 |

次のページに続く

表 311 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » cleared_alert_enable | boolean | false | none | 'クリアアラート' の通知をミュートするかどうかを決定します。 |
| » execute_script | boolean | false | none | アラートの状態が LOW/MEDIUM/HIGH に変更された場合に、提供されたスクリプトを実行するかどうかを決定します。 |
| » execute_script_on_clear | boolean | false | none | アラートの状態がクリアされた場合に、提供されたスクリプトを実行するかどうかを決定します。 |
| » execute_script_on_pem_serve | boolean | false | none | 提供されたスクリプトを PEM ホストまたはエージェントがオブジェクトを監視しているホストで実行するかどうかを決定します。 |
| » script_code | string | false | none | 実行するシェルまたはバッチスクリプト。詳細については、ドキュメントをご覧ください。 |
| » submit_to_nagios | boolean | false | none | アラートの状態が変更されたときに nagios に通知を送信するかどうかを決定します。 |
| » override_default_config | boolean | false | none | デフォルトの Webhook 構成をオーバーライドするかどうかを決定します。 |
| » send_notification | boolean | false | none | Webhook 送信通知を有効にするかどうかを決定します。 |
| » low_webhook_ids | [integer] | false | none | 低アラートが構成されている Webhook ID |
| » med_webhook_ids | [integer] | false | none | 中アラートが構成されている Webhook ID |
| » high_webhook_ids | [integer] | false | none | 高アラートが構成されている Webhook ID |
| » cleared_webhook_ids | [integer] | false | none | クリアされたアラートが構成されている Webhook ID |

### 列挙値

| Property | Value |
|---|---|
| paramname | Exclude tables smaller than |
| paramname | mount point |
| paramname | Host IP Address |
| paramname | Time interval to monitor |
| paramname | Time since idle |
| paramname | Per CPU utilization (K%) |

**注釈**

この操作には認証が必要ありません

## post_*alert_config_server*{server_id}*database*{database_name}*schema*{schema_name}*table*{table_name}

### コードサンプル

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/table/{table_name} \
  -H Content-Type: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Content-Type: application/json,
  X-Auth-Token: string
}

r = requests.post(https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/table/{table_name}, headers = headers)

print(r.json())
```

```
const inputBody = {
  "alert_name": "string",
  "alert_template": 0,
  "description": "string",
  "enabled": true,
  "history_retention": 0,
  "frequency_min": 0,
  "operator": "string",
```

(次のページに続く)

(前のページからの続き)

```
  "low_threshold_value": 0,
  "medium_threshold_value": 0,
  "high_threshold_value": 0,
  "all_alert_enable": true,
  "email_group_id": 1,
  "low_alert_enable": true,
  "low_email_group_id": 1,
  "med_alert_enable": true,
  "med_email_group_id": 1,
  "high_alert_enable": true,
  "high_email_group_id": 1,
  "send_trap": false,
  "snmp_trap_version": 0,
  "low_send_trap": true,
  "med_send_trap": true,
  "high_send_trap": true,
  "cleared_alert_enable": true,
  "params": [],
  "execute_script": true,
  "execute_script_on_clear": true,
  "execute_script_on_pem_server": true,
  "script_code": "string",
  "submit_to_nagios": true,
  "override_default_config": true,
  "send_notification": false,
  "low_webhook_ids": [],
  "med_webhook_ids": [],
  "high_webhook_ids": [],
  "cleared_webhook_ids": []
};
const headers = {
  Content-Type:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/database/
↪{database_name}/schema/{schema_name}/table/{table_name},
{
  method: POST,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
```

(次のページに続く)

(前のページからの続き)

```
    console.log(body);
});
```

POST /alert/config/server/{server_id}/database/{database_name}/schema/{schema_name}/table/{table_name}

テーブルレベルの Alert を作成します。

**ボディパラメーター**

```
{
  "alert_name": "string",
  "alert_template": 0,
  "description": "string",
  "enabled": true,
  "history_retention": 0,
  "frequency_min": 0,
  "operator": "string",
  "low_threshold_value": 0,
  "medium_threshold_value": 0,
  "high_threshold_value": 0,
  "all_alert_enable": true,
  "email_group_id": 1,
  "low_alert_enable": true,
  "low_email_group_id": 1,
  "med_alert_enable": true,
  "med_email_group_id": 1,
  "high_alert_enable": true,
  "high_email_group_id": 1,
  "send_trap": false,
  "snmp_trap_version": 0,
  "low_send_trap": true,
  "med_send_trap": true,
  "high_send_trap": true,
  "cleared_alert_enable": true,
  "params": [],
  "execute_script": true,
  "execute_script_on_clear": true,
  "execute_script_on_pem_server": true,
  "script_code": "string",
  "submit_to_nagios": true,
  "override_default_config": true,
  "send_notification": false,
  "low_webhook_ids": [],
  "med_webhook_ids": [],
  "high_webhook_ids": [],
```

(次のページに続く)

(前のページからの続き)

```
  "cleared_webhook_ids": []
}
```

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| database_name | path | string | true | データベース名 |
| schema_name | path | string | true | スキーマ名 |
| table_name | path | string | true | テーブル名 |
| X-Auth-Token | header | string | true | 承認トークン |
| body | body | *AlertPayload* | true | なし |

### 応答例

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ステータス 200 | None |

### レスポンススキーマ

**注釈**

この操作には認証が必要ありません

## get_*alert_config_server*{server_id}*database*{database_name}*schema*{schema_name}*table*{table_name}_{alert_i

### コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/table/{table_name}/{alert_id} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/table/{table_name}/{alert_id},␣
↪headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch(
  "https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/database/
↪{database_name}/schema/{schema_name}/table/{table_name}/{alert_id}",
  {
    method: "GET",

    headers: headers,
  },
)
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

`GET /alert/config/server/{server_id}/database/{database_name}/schema/{schema_name}/table/{table_name}/{alert_id}`

server_id、database_name、schema_name、table_name、および alert_id ごとにテーブルレベルの `Alerts` を取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| database_name | path | string | true | データベース名 |

次のページに続く

表 315 – 前のページからの続き

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| schema_name | path | string | true | スキーマ名 |
| table_name | path | string | true | テーブル名 |
| alert_id | path | integer | true | アラート ID |
| X-Auth-Token | header | string | true | 承認トークン |

**応答例**


```
{
  "id": 0,
  "alert_name": "string",
  "alert_template": 0,
  "description": "string",
  "enabled": true,
  "history_retention": 0,
  "frequency_min": 0,
  "auto_created": true,
  "template_id": "string",
  "params": [],
  "param_names": ["string"],
  "param_types": "string",
  "params_units": ["string"],
  "threshold_unit": "string",
  "operator": "string",
  "low_threshold_value": 0,
  "medium_threshold_value": 0,
  "high_threshold_value": 0,
  "send_email": true,
  "all_alert_enable": true,
  "email_group_id": 0,
  "low_alert_enable": true,
  "low_email_group_id": 0,
  "med_alert_enable": true,
  "med_email_group_id": 0,
  "high_alert_enable": true,
  "high_email_group_id": 0,
  "send_trap": true,
  "snmp_trap_version": 0,
  "low_send_trap": true,
  "med_send_trap": true,
  "high_send_trap": true,
  "cleared_alert_enable": true,
  "execute_script": true,
  "execute_script_on_clear": true,
```


(次のページに続く)

(前のページからの続き)

```
  "execute_script_on_pem_server": true,
  "script_code": "string",
  "submit_to_nagios": true,
  "override_default_config": true,
  "send_notification": true,
  "low_webhook_ids": [0],
  "med_webhook_ids": [0],
  "high_webhook_ids": [0],
  "cleared_webhook_ids": [0]
}
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | アラートスレッドについて |

**注釈**

この操作には認証が必要ありません

## put_*alert_config_server*{server_id}*database*{database_name}*schema*{schema_name}*table*{table_name}_{alert_i

### コードサンプル

```
#  You can also use wget

curl -X PUT https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/table/{table_name}/{alert_id} \
  -H Content-Type: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Content-Type: application/json,
  X-Auth-Token: string
}

r = requests.put(https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/table/{table_name}/{alert_id},␣
```

(次のページに続く)

(前のページからの続き)

```
↪headers = headers)

print(r.json())
```

```
const inputBody = {
  "alert_name": "string",
  "alert_template": 0,
  "description": "string",
  "enabled": true,
  "history_retention": 0,
  "frequency_min": 0,
  "operator": "string",
  "low_threshold_value": 0,
  "medium_threshold_value": 0,
  "high_threshold_value": 0,
  "all_alert_enable": true,
  "email_group_id": 1,
  "low_alert_enable": true,
  "low_email_group_id": 1,
  "med_alert_enable": true,
  "med_email_group_id": 1,
  "high_alert_enable": true,
  "high_email_group_id": 1,
  "send_trap": false,
  "snmp_trap_version": 0,
  "low_send_trap": true,
  "med_send_trap": true,
  "high_send_trap": true,
  "cleared_alert_enable": true,
  "params": [],
  "execute_script": true,
  "execute_script_on_clear": true,
  "execute_script_on_pem_server": true,
  "script_code": "string",
  "submit_to_nagios": true,
  "override_default_config": true,
  "send_notification": false,
  "low_webhook_ids": [],
  "med_webhook_ids": [],
  "high_webhook_ids": [],
  "cleared_webhook_ids": []
};
const headers = {
  Content-Type:application/json,
```

(次のページに続く)

(前のページからの続き)

```
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/database/
↪{database_name}/schema/{schema_name}/table/{table_name}/{alert_id},
{
  method: PUT,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

PUT /alert/config/server/{server_id}/database/{database_name}/schema/{schema_name}/table/{table_name}/{alert_id}

テーブルレベルの Alert を更新します。

**ボディパラメーター**

```
{
  "alert_name": "string",
  "alert_template": 0,
  "description": "string",
  "enabled": true,
  "history_retention": 0,
  "frequency_min": 0,
  "operator": "string",
  "low_threshold_value": 0,
  "medium_threshold_value": 0,
  "high_threshold_value": 0,
  "all_alert_enable": true,
  "email_group_id": 1,
  "low_alert_enable": true,
  "low_email_group_id": 1,
  "med_alert_enable": true,
  "med_email_group_id": 1,
  "high_alert_enable": true,
  "high_email_group_id": 1,
  "send_trap": false,
  "snmp_trap_version": 0,
  "low_send_trap": true,
  "med_send_trap": true,
```

(次のページに続く)

(前のページからの続き)


```
  "high_send_trap": true,
  "cleared_alert_enable": true,
  "params": [],
  "execute_script": true,
  "execute_script_on_clear": true,
  "execute_script_on_pem_server": true,
  "script_code": "string",
  "submit_to_nagios": true,
  "override_default_config": true,
  "send_notification": false,
  "low_webhook_ids": [],
  "med_webhook_ids": [],
  "high_webhook_ids": [],
  "cleared_webhook_ids": []
}
```


### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| database_name | path | string | true | データベース名 |
| schema_name | path | string | true | スキーマ名 |
| table_name | path | string | true | テーブル名 |
| alert_id | path | integer | true | アラート ID |
| X-Auth-Token | header | string | true | 承認トークン |
| body | body | *AlertPayload* | true | なし |

### 応答例

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ステータス 200 | None |

レスポンススキーマ

> **注釈**
>
> この操作には認証が必要ありません

**delete_*alert_config_server*{server_id}*database*{database_name}*schema*{schema_name}*table*{table_name}_{aler**

コードサンプル

```
#  You can also use wget

curl -X DELETE https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/table/{table_name}/{alert_id} \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  X-Auth-Token: string
}

r = requests.delete(https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/table/{table_name}/{alert_id},␣
↪headers = headers)

print(r.json())
```

```
const headers = {
  "X-Auth-Token": "string",
};

fetch(
  "https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/database/
↪{database_name}/schema/{schema_name}/table/{table_name}/{alert_id}",
  {
    method: "DELETE",

    headers: headers,
  },
)
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

`DELETE /alert/config/server/{server_id}/database/{database_name}/schema/{schema_name}/table/{table_name}/{alert_id}`

`Alert` オブジェクトを削除します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| database_name | path | string | true | データベース名 |
| schema_name | path | string | true | スキーマ名 |
| table_name | path | string | true | テーブル名 |
| alert_id | path | integer | true | アラート ID |
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 削除が成功しました。 | None |

### レスポンススキーマ

**注釈**

この操作には認証が必要ありません

## get_*alert_config_server*{server_id}*database*{database_name}*schema*{schema_name}*index*{index_name}

### コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/index/{index_name} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
```

(次のページに続く)

(前のページからの続き)

```
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/index/{index_name}, headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch(
  "https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/database/
↪{database_name}/schema/{schema_name}/index/{index_name}",
  {
    method: "GET",

    headers: headers,
  },
)
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

`GET /alert/config/server/{server_id}/database/{database_name}/schema/{schema_name}/index/{index_name}`

server_id、database_name、schema_name、および index_name ごとにインデックスレベルの `Alerts` を取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| database_name | path | string | true | データベース名 |
| schema_name | path | string | true | スキーマ名 |
| index_name | path | string | true | インデックス名 |

次のページに続く

表 321 – 前のページからの続き

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| X-Auth-Token | header | string | true | 承認トークン |

**応答例**


```
[
  {
    "id": 0,
    "alert_name": "string",
    "alert_template": 0,
    "description": "string",
    "enabled": true,
    "history_retention": 0,
    "frequency_min": 0,
    "auto_created": true,
    "template_id": "string",
    "params": [],
    "param_names": ["string"],
    "param_types": "string",
    "params_units": ["string"],
    "threshold_unit": "string",
    "operator": "string",
    "low_threshold_value": 0,
    "medium_threshold_value": 0,
    "high_threshold_value": 0,
    "send_email": true,
    "all_alert_enable": true,
    "email_group_id": 0,
    "low_alert_enable": true,
    "low_email_group_id": 0,
    "med_alert_enable": true,
    "med_email_group_id": 0,
    "high_alert_enable": true,
    "high_email_group_id": 0,
    "send_trap": true,
    "snmp_trap_version": 0,
    "low_send_trap": true,
    "med_send_trap": true,
    "high_send_trap": true,
    "cleared_alert_enable": true,
    "execute_script": true,
    "execute_script_on_clear": true,
    "execute_script_on_pem_server": true,
    "script_code": "string",
```


(次のページに続く)

(前のページからの続き)


```
    "submit_to_nagios": true,
    "override_default_config": true,
    "send_notification": true,
    "low_webhook_ids": [0],
    "med_webhook_ids": [0],
    "high_webhook_ids": [0],
    "cleared_webhook_ids": [0]
  }
]
```


### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | Inline |

### レスポンススキーマ

ステータスコード **200**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| _anonymous_ | *Platform Compatibility* ] | false | none | なし |
| » id | integer | false | none | アラート ID。注このプロパティは、POST 要求には適用されません。 |
| » alert_name | string | false | none | アラートの名前。注 このプロパティは、POST 要求の必須パラメーターです。 |
| » alert_template | integer | false | none | アラートテンプレートの ID。このプロパティは、正の整数値である必要があります。注これは POST 要求の必須パラメーターです。 |
| » description | string | false | none | なし |

次のページに続く

表 323 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » enabled | boolean | false | none | このアラートが有効/無効かどうかを判断します。注 このプロパティは、POST 要求の必須パラメーターです。 |
| » history_retention | integer | false | none | このアラートの履歴を保持する日数を指定します。指定できる値は 1 ～99999 です。 注このプロパティは、POST 要求の必須パラメーターです。 |
| » frequency_min | integer | false | none | PEM データベースサーバーでアラートを確認する頻度を分単位で定義します。指定できる値は 1 ～65534 です。 注このプロパティは、POST 要求の必須パラメーターです。 |
| » auto_created | boolean | false | none | アラートを自動作成するかどうかを指定します。 |
| » template_id | string | false | none | アラートテンプレートを定義します。 |
| » params | *Platform Compatibility* ] | false | none | アラートを確認するためにアラートテンプレートが使用するパラメーター/値のペアのリスト。注 アラートテンプレートには、all パラメーターを定義する必要があります。 |
| » paramname | string | false | none | パラメーターの名前。注このプロパティは、アラートテンプレートパラメーターリストに存在する必要があります。例 |
| » paramvalue | string | false | none | 指定されたパラメーターの値 |
| » param_names | [string] | false | none | パラメーターの名前を定義します。 |
| » param_types | string | false | none | パラメーターのタイプを定義します。例 INTEGER/STRING.. |

次のページに続く

表 323 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » params_units | [string] | false | none | パラメーターに使用される単位を定義します。例 'DAYS'、'True or False'、'Hours'、'Minutes' など |
| » threshold_unit | string | false | none | MB/kB/秒/日/... などの、しきい値の計算に使用される単位 この値は、GUI でのみ使用されます。 |
| » operator | string | false | none | しきい値をチェックするための演算子を定義します。有効な文字列は>"および"<"です。注 このプロパティは、POST 要求の必須パラメーターです。 |
| » low_threshold_value | number | false | none | 低アラートをトリガーするためのしきい値。このプロパティには、有効な整数/浮動小数点値を指定できます。注これは POST 要求の必須パラメーターです。 |
| » medium_threshold_value | number | false | none | 中アラートをトリガーするためのしきい値。このプロパティには、有効な整数/浮動小数点値を指定できます。注これは POST 要求の必須パラメーターです。 |
| » high_threshold_value | number | false | none | 高度なアラートをトリガーするためのしきい値。任意の有効な整数/浮動小数点値を指定できます。注これは POST 要求の必須パラメーターです。 |
| » send_email | boolean | false | none | アラートの状態が変更されたときに電子メールを送信するかどうかを決定します。 |

次のページに続く

表 323 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » all_alert_enable | boolean | false | none | アラートの状態が変更されたときに、パラメーター 'email_group_id' で指定されたグループにメール通知を送信するかどうかを決定します。 |
| » email_group_id | integer | false | none | アラートの状態が変更されたときに電子メール通知の送信先の電子メールグループを定義します。 |
| » low_alert_enable | boolean | false | none | アラートの状態が LOW に設定されているときに、パラメーター 'low_email_group_id' で指定されたグループにメール通知を送信するかどうかを決定します。 |
| » low_email_group_id | integer | false | none | アラートの状態が LOW に変更されたときに電子メール通知の送信先の電子メールグループを定義します。 |
| » med_alert_enable | boolean | false | none | アラートの状態が MEDIUM に設定されている場合、パラメーター 'medium_email_group_id' で指定されたグループにメール通知を送信するかどうかを決定します。 |
| » med_email_group_id | integer | false | none | アラートの状態が MEDIUM に変更されたときにメール通知の送信先のメールグループを定義します。 |
| » high_alert_enable | boolean | false | none | アラートの状態が HIGH に設定されているときに、パラメーター 'high_email_group_id' で指定されたグループに電子メール通知を送信するかどうかを決定します。 |

次のページに続く

表 323 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » high_email_group_id | integer | false | none | アラートの状態が HIGH に変更されたときに電子メール通知の送信先の電子メールグループを定義します。 |
| » send_trap | boolean | false | none | アラートの状態が変更されたときに SNMP トラップ通知を送信するかどうかを決定します。 |
| » snmp_trap_version | integer | false | none | SNMP トラップバージョンを定義します。サポートされている SNMP トラップバージョンは 1、2、または 3 です |
| » low_send_trap | boolean | false | none | アラートの状態が LOW に変更されたときに SNMP トラップ通知を送信するかどうかを決定します。 |
| » med_send_trap | boolean | false | none | アラートの状態が MEDIUM に変更されたときに SNMP トラップ通知を送信するかどうかを決定します。 |
| » high_send_trap | boolean | false | none | アラートの状態が HIGH に変わったときに SNMP トラップ通知を送信するかどうかを決定します。 |
| » cleared_alert_enable | boolean | false | none | ' クリアアラート' の通知をミュートするかどうかを決定します。 |
| » execute_script | boolean | false | none | アラートの状態が LOW/MEDIUM/HIGH に変更された場合に、提供されたスクリプトを実行するかどうかを決定します。 |
| » execute_script_on_clear | boolean | false | none | アラートの状態がクリアされた場合に、提供されたスクリプトを実行するかどうかを決定します。 |

次のページに続く

表 323 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » execute_script_on_pem_serve | boolean | false | none | 提供されたスクリプトをPEM ホストまたはエージェントがオブジェクトを監視しているホストで実行するかどうかを決定します。 |
| » script_code | string | false | none | 実行するシェルまたはバッチスクリプト。詳細については、ドキュメントをご覧ください。 |
| » submit_to_nagios | boolean | false | none | アラートの状態が変更されたときに nagios に通知を送信するかどうかを決定します。 |
| » override_default_config | boolean | false | none | デフォルトの Webhook 構成をオーバーライドするかどうかを決定します。 |
| » send_notification | boolean | false | none | Webhook 送信通知を有効にするかどうかを決定します。 |
| » low_webhook_ids | [integer] | false | none | 低アラートが構成されている Webhook ID |
| » med_webhook_ids | [integer] | false | none | 中アラートが構成されている Webhook ID |
| » high_webhook_ids | [integer] | false | none | 高アラートが構成されている Webhook ID |
| » cleared_webhook_ids | [integer] | false | none | クリアされたアラートが構成されている Webhook ID |

### 列挙値

| Property | Value |
|---|---|
| paramname | Exclude tables smaller than |
| paramname | mount point |
| paramname | Host IP Address |
| paramname | Time interval to monitor |
| paramname | Time since idle |
| paramname | Per CPU utilization (K%) |

> **注釈**
>
> この操作には認証が必要ありません

### post_*alert_config_server*{server_id}*database*{database_name}*schema*{schema_name}*index*{index_name}

#### コードサンプル

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/index/{index_name} \
  -H Content-Type: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Content-Type: application/json,
  X-Auth-Token: string
}

r = requests.post(https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/index/{index_name}, headers = headers)

print(r.json())
```

```
const inputBody = {
  "alert_name": "string",
  "alert_template": 0,
  "description": "string",
  "enabled": true,
  "history_retention": 0,
  "frequency_min": 0,
  "operator": "string",
  "low_threshold_value": 0,
  "medium_threshold_value": 0,
  "high_threshold_value": 0,
  "all_alert_enable": true,
  "email_group_id": 1,
  "low_alert_enable": true,
  "low_email_group_id": 1,
  "med_alert_enable": true,
  "med_email_group_id": 1,
  "high_alert_enable": true,
  "high_email_group_id": 1,
```

(次のページに続く)

(前のページからの続き)

```
  "send_trap": false,
  "snmp_trap_version": 0,
  "low_send_trap": true,
  "med_send_trap": true,
  "high_send_trap": true,
  "cleared_alert_enable": true,
  "params": [],
  "execute_script": true,
  "execute_script_on_clear": true,
  "execute_script_on_pem_server": true,
  "script_code": "string",
  "submit_to_nagios": true,
  "override_default_config": true,
  "send_notification": false,
  "low_webhook_ids": [],
  "med_webhook_ids": [],
  "high_webhook_ids": [],
  "cleared_webhook_ids": []
};
const headers = {
  Content-Type:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/database/
↪{database_name}/schema/{schema_name}/index/{index_name},
{
  method: POST,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

POST /alert/config/server/{server_id}/database/{database_name}/schema/{schema_name}/index/{index_name}

インデックスレベルの Alert を作成します。

**ボディパラメーター**

```
{
  "alert_name": "string",
```

(次のページに続く)

(前のページからの続き)


```
  "alert_template": 0,
  "description": "string",
  "enabled": true,
  "history_retention": 0,
  "frequency_min": 0,
  "operator": "string",
  "low_threshold_value": 0,
  "medium_threshold_value": 0,
  "high_threshold_value": 0,
  "all_alert_enable": true,
  "email_group_id": 1,
  "low_alert_enable": true,
  "low_email_group_id": 1,
  "med_alert_enable": true,
  "med_email_group_id": 1,
  "high_alert_enable": true,
  "high_email_group_id": 1,
  "send_trap": false,
  "snmp_trap_version": 0,
  "low_send_trap": true,
  "med_send_trap": true,
  "high_send_trap": true,
  "cleared_alert_enable": true,
  "params": [],
  "execute_script": true,
  "execute_script_on_clear": true,
  "execute_script_on_pem_server": true,
  "script_code": "string",
  "submit_to_nagios": true,
  "override_default_config": true,
  "send_notification": false,
  "low_webhook_ids": [],
  "med_webhook_ids": [],
  "high_webhook_ids": [],
  "cleared_webhook_ids": []
}
```


### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| database_name | path | string | true | データベース名 |
| schema_name | path | string | true | スキーマ名 |

次のページに続く

表 325 – 前のページからの続き

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| index_name | path | string | true | インデックス名 |
| X-Auth-Token | header | string | true | 承認トークン |
| body | body | *AlertPayload* | true | なし |

**応答例**

**応答**

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ステータス 200 | None |

**レスポンススキーマ**

**注釈**

この操作には認証が必要ありません

**get_*alert_config_server*{server_id}*database*{database_name}*schema*{schema_name}*index*{index_name}_{alert_**

**コードサンプル**

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/index/{index_name}/{alert_id} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/index/{index_name}/{alert_id},␣
↪headers = headers)
```

(次のページに続く)

(前のページからの続き)

```
print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch(
  "https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/database/
↪{database_name}/schema/{schema_name}/index/{index_name}/{alert_id}",
  {
    method: "GET",

    headers: headers,
  },
)
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

`GET /alert/config/server/{server_id}/database/{database_name}/schema/{schema_name}/index/{index_name}/{alert_id}`

server_id、database_name、schema_name、index_name、および alert_id ごとにインデックスレベルの `Alerts` を取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| database_name | path | string | true | データベース名 |
| schema_name | path | string | true | スキーマ名 |
| index_name | path | string | true | インデックス名 |
| alert_id | path | integer | true | アラート ID |
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例

```
{
  "id": 0,
```

(次のページに続く)

(前のページからの続き)


```
  "alert_name": "string",
  "alert_template": 0,
  "description": "string",
  "enabled": true,
  "history_retention": 0,
  "frequency_min": 0,
  "auto_created": true,
  "template_id": "string",
  "params": [],
  "param_names": ["string"],
  "param_types": "string",
  "params_units": ["string"],
  "threshold_unit": "string",
  "operator": "string",
  "low_threshold_value": 0,
  "medium_threshold_value": 0,
  "high_threshold_value": 0,
  "send_email": true,
  "all_alert_enable": true,
  "email_group_id": 0,
  "low_alert_enable": true,
  "low_email_group_id": 0,
  "med_alert_enable": true,
  "med_email_group_id": 0,
  "high_alert_enable": true,
  "high_email_group_id": 0,
  "send_trap": true,
  "snmp_trap_version": 0,
  "low_send_trap": true,
  "med_send_trap": true,
  "high_send_trap": true,
  "cleared_alert_enable": true,
  "execute_script": true,
  "execute_script_on_clear": true,
  "execute_script_on_pem_server": true,
  "script_code": "string",
  "submit_to_nagios": true,
  "override_default_config": true,
  "send_notification": true,
  "low_webhook_ids": [0],
  "med_webhook_ids": [0],
  "high_webhook_ids": [0],
  "cleared_webhook_ids": [0]
}
```

応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | アラートスレッドについて |

**注釈**

この操作には認証が必要ありません

**put_*alert_config_server*{server_id}*database*{database_name}*schema*{schema_name}*index*{index_name}_{alert_**

コードサンプル

```
#  You can also use wget

curl -X PUT https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/index/{index_name}/{alert_id} \
  -H Content-Type: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Content-Type: application/json,
  X-Auth-Token: string
}

r = requests.put(https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/index/{index_name}/{alert_id},␣
↪headers = headers)

print(r.json())
```

```
const inputBody = {
  "alert_name": "string",
  "alert_template": 0,
  "description": "string",
  "enabled": true,
  "history_retention": 0,
  "frequency_min": 0,
  "operator": "string",
  "low_threshold_value": 0,
```

(次のページに続く)

(前のページからの続き)

```
  "medium_threshold_value": 0,
  "high_threshold_value": 0,
  "all_alert_enable": true,
  "email_group_id": 1,
  "low_alert_enable": true,
  "low_email_group_id": 1,
  "med_alert_enable": true,
  "med_email_group_id": 1,
  "high_alert_enable": true,
  "high_email_group_id": 1,
  "send_trap": false,
  "snmp_trap_version": 0,
  "low_send_trap": true,
  "med_send_trap": true,
  "high_send_trap": true,
  "cleared_alert_enable": true,
  "params": [],
  "execute_script": true,
  "execute_script_on_clear": true,
  "execute_script_on_pem_server": true,
  "script_code": "string",
  "submit_to_nagios": true,
  "override_default_config": true,
  "send_notification": false,
  "low_webhook_ids": [],
  "med_webhook_ids": [],
  "high_webhook_ids": [],
  "cleared_webhook_ids": []
};
const headers = {
  Content-Type:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/database/
↪{database_name}/schema/{schema_name}/index/{index_name}/{alert_id},
{
  method: PUT,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
```

(次のページに続く)

(前のページからの続き)

```
});
```

PUT /alert/config/server/{server_id}/database/{database_name}/schema/{schema_name}/index/{index_name}/{alert_id}

インデックスレベルの Alert を更新します。

**ボディパラメーター**

```
{
  "alert_name": "string",
  "alert_template": 0,
  "description": "string",
  "enabled": true,
  "history_retention": 0,
  "frequency_min": 0,
  "operator": "string",
  "low_threshold_value": 0,
  "medium_threshold_value": 0,
  "high_threshold_value": 0,
  "all_alert_enable": true,
  "email_group_id": 1,
  "low_alert_enable": true,
  "low_email_group_id": 1,
  "med_alert_enable": true,
  "med_email_group_id": 1,
  "high_alert_enable": true,
  "high_email_group_id": 1,
  "send_trap": false,
  "snmp_trap_version": 0,
  "low_send_trap": true,
  "med_send_trap": true,
  "high_send_trap": true,
  "cleared_alert_enable": true,
  "params": [],
  "execute_script": true,
  "execute_script_on_clear": true,
  "execute_script_on_pem_server": true,
  "script_code": "string",
  "submit_to_nagios": true,
  "override_default_config": true,
  "send_notification": false,
  "low_webhook_ids": [],
  "med_webhook_ids": [],
  "high_webhook_ids": [],
  "cleared_webhook_ids": []
```

(次のページに続く)

(前のページからの続き)

```
}
```

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| database_name | path | string | true | データベース名 |
| schema_name | path | string | true | スキーマ名 |
| index_name | path | string | true | インデックス名 |
| alert_id | path | integer | true | アラート ID |
| X-Auth-Token | header | string | true | 承認トークン |
| body | body | *AlertPayload* | true | なし |

### 応答例

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ステータス 200 | None |

### レスポンススキーマ

**注釈**

この操作には認証が必要ありません

## delete_*alert_config_server*{server_id}*database*{database_name}*schema*{schema_name}*index*{index_name}_{ale

### コードサンプル

```
#  You can also use wget

curl -X DELETE https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/index/{index_name}/{alert_id} \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  X-Auth-Token: string
}

r = requests.delete(https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/index/{index_name}/{alert_id},␣
↪headers = headers)

print(r.json())
```

```
const headers = {
  "X-Auth-Token": "string",
};

fetch(
  "https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/database/
↪{database_name}/schema/{schema_name}/index/{index_name}/{alert_id}",
  {
    method: "DELETE",

    headers: headers,
  },
)
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

`DELETE /alert/config/server/{server_id}/database/{database_name}/schema/{schema_name}/index/{index_name}/{alert_id}`

`Alert` オブジェクトを削除します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| database_name | path | string | true | データベース名 |
| schema_name | path | string | true | スキーマ名 |
| index_name | path | string | true | インデックス名 |
| alert_id | path | integer | true | アラート ID |

次のページに続く

表 331 – 前のページからの続き

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例

#### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 削除が成功しました。 | None |

#### レスポンススキーマ

**注釈**

この操作には認証が必要ありません

### get_*alert_config_server*{server_id}*database*{database_name}*schema*{schema_name}*sequence*{sequence_name

### コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/sequence/{sequence_name} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/sequence/{sequence_name}, headers =␣
↪headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch(
  "https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/database/
↪{database_name}/schema/{schema_name}/sequence/{sequence_name}",
  {
    method: "GET",

    headers: headers,
  },
)
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

`GET /alert/config/server/{server_id}/database/{database_name}/schema/{schema_name}/sequence/{sequence_name}`

server_id、database_name、schema_name、および sequence_name ごとにシーケンスレベルの `Alerts` を取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| database_name | path | string | true | データベース名 |
| schema_name | path | string | true | スキーマ名 |
| sequence_name | path | string | true | シークエンス名 |
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例

```
[
  {
    "id": 0,
    "alert_name": "string",
    "alert_template": 0,
    "description": "string",
    "enabled": true,
```

(次のページに続く)

(前のページからの続き)


```
    "history_retention": 0,
    "frequency_min": 0,
    "auto_created": true,
    "template_id": "string",
    "params": [],
    "param_names": ["string"],
    "param_types": "string",
    "params_units": ["string"],
    "threshold_unit": "string",
    "operator": "string",
    "low_threshold_value": 0,
    "medium_threshold_value": 0,
    "high_threshold_value": 0,
    "send_email": true,
    "all_alert_enable": true,
    "email_group_id": 0,
    "low_alert_enable": true,
    "low_email_group_id": 0,
    "med_alert_enable": true,
    "med_email_group_id": 0,
    "high_alert_enable": true,
    "high_email_group_id": 0,
    "send_trap": true,
    "snmp_trap_version": 0,
    "low_send_trap": true,
    "med_send_trap": true,
    "high_send_trap": true,
    "cleared_alert_enable": true,
    "execute_script": true,
    "execute_script_on_clear": true,
    "execute_script_on_pem_server": true,
    "script_code": "string",
    "submit_to_nagios": true,
    "override_default_config": true,
    "send_notification": true,
    "low_webhook_ids": [0],
    "med_webhook_ids": [0],
    "high_webhook_ids": [0],
    "cleared_webhook_ids": [0]
  }
]
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | Inline |

### レスポンススキーマ

ステータスコード **200**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| _anonymous_ | *Platform Compatibility* ] | false | none | なし |
| » id | integer | false | none | アラート ID。注このプロパティは、POST 要求には適用されません。 |
| » alert_name | string | false | none | アラートの名前。注 このプロパティは、POST 要求の必須パラメーターです。 |
| » alert_template | integer | false | none | アラートテンプレートの ID。このプロパティは、正の整数値である必要があります。注これはPOST 要求の必須パラメーターです。 |
| » description | string | false | none | なし |
| » enabled | boolean | false | none | このアラートが有効/無効かどうかを判断します。注 このプロパティは、POST 要求の必須パラメーターです。 |
| » history_retention | integer | false | none | このアラートの履歴を保持する日数を指定します。指定できる値は 1 ～99999 です。 注このプロパティは、POST 要求の必須パラメーターです。 |

次のページに続く

表 335 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » frequency_min | integer | false | none | PEM データベースサーバーでアラートを確認する頻度を分単位で定義します。指定できる値は 1〜65534 です。 注このプロパティは、POST 要求の必須パラメーターです。 |
| » auto_created | boolean | false | none | アラートを自動作成するかどうかを指定します。 |
| » template_id | string | false | none | アラートテンプレートを定義します。 |
| » params | *Platform Compatibility* ] | false | none | アラートを確認するためにアラートテンプレートが使用するパラメーター/値のペアのリスト。注アラートテンプレートには、all パラメーターを定義する必要があります。 |
| » paramname | string | false | none | パラメーターの名前。注このプロパティは、アラートテンプレートパラメーターリストに存在する必要があります。例 |
| » paramvalue | string | false | none | 指定されたパラメーターの値 |
| » param_names | [string] | false | none | パラメーターの名前を定義します。 |
| » param_types | string | false | none | パラメーターのタイプを定義します。例 INTEGER/STRING.. |
| » params_units | [string] | false | none | パラメーターに使用される単位を定義します。例 'DAYS'、'True or False'、'Hours'、'Minutes' など |
| » threshold_unit | string | false | none | MB/kB/秒/日/... などの、しきい値の計算に使用される単位 この値は、GUI でのみ使用されます。 |

次のページに続く

表 335 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » operator | string | false | none | しきい値をチェックするための演算子を定義します。有効な文字列は>"および"<"です。注このプロパティは、POST要求の必須パラメーターです。 |
| » low_threshold_value | number | false | none | 低アラートをトリガーするためのしきい値。このプロパティには、有効な整数/浮動小数点値を指定できます。注これはPOST要求の必須パラメーターです。 |
| » medium_threshold_value | number | false | none | 中アラートをトリガーするためのしきい値。このプロパティには、有効な整数/浮動小数点値を指定できます。注これはPOST要求の必須パラメーターです。 |
| » high_threshold_value | number | false | none | 高度なアラートをトリガーするためのしきい値。任意の有効な整数/浮動小数点値を指定できます。注これはPOST要求の必須パラメーターです。 |
| » send_email | boolean | false | none | アラートの状態が変更されたときに電子メールを送信するかどうかを決定します。 |
| » all_alert_enable | boolean | false | none | アラートの状態が変更されたときに、パラメーター 'email_group_id' で指定されたグループにメール通知を送信するかどうかを決定します。 |
| » email_group_id | integer | false | none | アラートの状態が変更されたときに電子メール通知の送信先の電子メールグループを定義します。 |

次のページに続く

表 335 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » low_alert_enable | boolean | false | none | アラートの状態が LOW に設定されているときに、パラメーター 'low_email_group_id' で指定されたグループにメール通知を送信するかどうかを決定します。 |
| » low_email_group_id | integer | false | none | アラートの状態が LOW に変更されたときに電子メール通知の送信先の電子メールグループを定義します。 |
| » med_alert_enable | boolean | false | none | アラートの状態が MEDIUM に設定されている場合、パラメーター 'medium_email_group_id' で指定されたグループにメール通知を送信するかどうかを決定します。 |
| » med_email_group_id | integer | false | none | アラートの状態が MEDIUM に変更されたときにメール通知の送信先のメールグループを定義します。 |
| » high_alert_enable | boolean | false | none | アラートの状態が HIGH に設定されているときに、パラメーター 'high_email_group_id' で指定されたグループに電子メール通知を送信するかどうかを決定します。 |
| » high_email_group_id | integer | false | none | アラートの状態が HIGH に変更されたときに電子メール通知の送信先の電子メールグループを定義します。 |
| » send_trap | boolean | false | none | アラートの状態が変更されたときに SNMP トラップ通知を送信するかどうかを決定します。 |

次のページに続く

表 335 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » snmp_trap_version | integer | false | none | SNMP トラップバージョンを定義します。サポートされているSNMP トラップバージョンは 1、2、または 3 です |
| » low_send_trap | boolean | false | none | アラートの状態が LOW に変更されたときに SNMP トラップ通知を送信するかどうかを決定します。 |
| » med_send_trap | boolean | false | none | アラートの状態が MEDIUM に変更されたときに SNMP トラップ通知を送信するかどうかを決定します。 |
| » high_send_trap | boolean | false | none | アラートの状態が HIGH に変わったときに SNMP トラップ通知を送信するかどうかを決定します。 |
| » cleared_alert_enable | boolean | false | none | ' クリアアラート' の通知をミュートするかどうかを決定します。 |
| » execute_script | boolean | false | none | アラートの状態が LOW/MEDIUM/HIGH に変更された場合に、提供されたスクリプトを実行するかどうかを決定します。 |
| » execute_script_on_clear | boolean | false | none | アラートの状態がクリアされた場合に、提供されたスクリプトを実行するかどうかを決定します。 |
| » execute_script_on_pem_serve | boolean | false | none | 提供されたスクリプトを PEM ホストまたはエージェントがオブジェクトを監視しているホストで実行するかどうかを決定します。 |

次のページに続く

表 335 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » script_code | string | false | none | 実行するシェルまたはバッチスクリプト。詳細については、ドキュメントをご覧ください。 |
| » submit_to_nagios | boolean | false | none | アラートの状態が変更されたときに nagios に通知を送信するかどうかを決定します。 |
| » override_default_config | boolean | false | none | デフォルトの Webhook 構成をオーバーライドするかどうかを決定します。 |
| » send_notification | boolean | false | none | Webhook 送信通知を有効にするかどうかを決定します。 |
| » low_webhook_ids | [integer] | false | none | 低アラートが構成されている Webhook ID |
| » med_webhook_ids | [integer] | false | none | 中アラートが構成されている Webhook ID |
| » high_webhook_ids | [integer] | false | none | 高アラートが構成されている Webhook ID |
| » cleared_webhook_ids | [integer] | false | none | クリアされたアラートが構成されている Webhook ID |

### 列挙値

| Property | Value |
|---|---|
| paramname | Exclude tables smaller than |
| paramname | mount point |
| paramname | Host IP Address |
| paramname | Time interval to monitor |
| paramname | Time since idle |
| paramname | Per CPU utilization (K%) |

**注釈**

この操作には認証が必要ありません

### post_*alert_config_server*{server_id}*database*{database_name}*schema*{schema_name}*sequence*{sequence_nam

コードサンプル

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/sequence/{sequence_name} \
  -H Content-Type: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Content-Type: application/json,
  X-Auth-Token: string
}

r = requests.post(https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/sequence/{sequence_name}, headers =␣
↪headers)

print(r.json())
```

```
const inputBody = {
  "alert_name": "string",
  "alert_template": 0,
  "description": "string",
  "enabled": true,
  "history_retention": 0,
  "frequency_min": 0,
  "operator": "string",
  "low_threshold_value": 0,
  "medium_threshold_value": 0,
  "high_threshold_value": 0,
  "all_alert_enable": true,
  "email_group_id": 1,
  "low_alert_enable": true,
  "low_email_group_id": 1,
  "med_alert_enable": true,
  "med_email_group_id": 1,
  "high_alert_enable": true,
  "high_email_group_id": 1,
  "send_trap": false,
  "snmp_trap_version": 0,
  "low_send_trap": true,
  "med_send_trap": true,
```

(次のページに続く)

(前のページからの続き)

```
  "high_send_trap": true,
  "cleared_alert_enable": true,
  "params": [],
  "execute_script": true,
  "execute_script_on_clear": true,
  "execute_script_on_pem_server": true,
  "script_code": "string",
  "submit_to_nagios": true,
  "override_default_config": true,
  "send_notification": false,
  "low_webhook_ids": [],
  "med_webhook_ids": [],
  "high_webhook_ids": [],
  "cleared_webhook_ids": []
};
const headers = {
  Content-Type:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/database/
↪{database_name}/schema/{schema_name}/sequence/{sequence_name},
{
  method: POST,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

`POST /alert/config/server/{server_id}/database/{database_name}/schema/{schema_name}/sequence/{sequence_name}`

シーケンスレベルの `Alert` を作成します。

**ボディパラメーター**

```
{
  "alert_name": "string",
  "alert_template": 0,
  "description": "string",
  "enabled": true,
  "history_retention": 0,
```

(次のページに続く)

(前のページからの続き)


```
  "frequency_min": 0,
  "operator": "string",
  "low_threshold_value": 0,
  "medium_threshold_value": 0,
  "high_threshold_value": 0,
  "all_alert_enable": true,
  "email_group_id": 1,
  "low_alert_enable": true,
  "low_email_group_id": 1,
  "med_alert_enable": true,
  "med_email_group_id": 1,
  "high_alert_enable": true,
  "high_email_group_id": 1,
  "send_trap": false,
  "snmp_trap_version": 0,
  "low_send_trap": true,
  "med_send_trap": true,
  "high_send_trap": true,
  "cleared_alert_enable": true,
  "params": [],
  "execute_script": true,
  "execute_script_on_clear": true,
  "execute_script_on_pem_server": true,
  "script_code": "string",
  "submit_to_nagios": true,
  "override_default_config": true,
  "send_notification": false,
  "low_webhook_ids": [],
  "med_webhook_ids": [],
  "high_webhook_ids": [],
  "cleared_webhook_ids": []
}
```


### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| database_name | path | string | true | データベース名 |
| schema_name | path | string | true | スキーマ名 |
| sequence_name | path | string | true | シークエンス名 |
| X-Auth-Token | header | string | true | 承認トークン |
| body | body | *AlertPayload* | true | なし |

### 応答例

#### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ステータス 200 | None |

#### レスポンススキーマ

> **注釈**
>
> この操作には認証が必要ありません

### get_*alert_config_server*{server_id}*database*{database_name}*schema*{schema_name}*sequence*{sequence_name

#### コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/sequence/{sequence_name}/{alert_id} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/sequence/{sequence_name}/{alert_id},␣
↪headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
```

(次のページに続く)

(前のページからの続き)

```
};

fetch(
  "https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/database/
↪{database_name}/schema/{schema_name}/sequence/{sequence_name}/{alert_id}",
  {
    method: "GET",

    headers: headers,
  },
)
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

`GET /alert/config/server/{server_id}/database/{database_name}/schema/{schema_name}/sequence/{sequence_name}/{alert_id}`

server_id、database_name、schema_name、sequence_name、および alert_id ごとにシーケンスレベルの `Alerts` を取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| database_name | path | string | true | データベース名 |
| schema_name | path | string | true | スキーマ名 |
| sequence_name | path | string | true | シークエンス名 |
| alert_id | path | integer | true | アラート ID |
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例

```
{
  "id": 0,
  "alert_name": "string",
  "alert_template": 0,
  "description": "string",
  "enabled": true,
  "history_retention": 0,
  "frequency_min": 0,
```

(次のページに続く)

(前のページからの続き)

```
  "auto_created": true,
  "template_id": "string",
  "params": [],
  "param_names": ["string"],
  "param_types": "string",
  "params_units": ["string"],
  "threshold_unit": "string",
  "operator": "string",
  "low_threshold_value": 0,
  "medium_threshold_value": 0,
  "high_threshold_value": 0,
  "send_email": true,
  "all_alert_enable": true,
  "email_group_id": 0,
  "low_alert_enable": true,
  "low_email_group_id": 0,
  "med_alert_enable": true,
  "med_email_group_id": 0,
  "high_alert_enable": true,
  "high_email_group_id": 0,
  "send_trap": true,
  "snmp_trap_version": 0,
  "low_send_trap": true,
  "med_send_trap": true,
  "high_send_trap": true,
  "cleared_alert_enable": true,
  "execute_script": true,
  "execute_script_on_clear": true,
  "execute_script_on_pem_server": true,
  "script_code": "string",
  "submit_to_nagios": true,
  "override_default_config": true,
  "send_notification": true,
  "low_webhook_ids": [0],
  "med_webhook_ids": [0],
  "high_webhook_ids": [0],
  "cleared_webhook_ids": [0]
}
```

応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | アラートスレッドについて |

**注釈**

この操作には認証が必要ありません

### put_*alert_config_server*{server_id}*database*{database_name}*schema*{schema_name}*sequence*{sequence_name

コードサンプル

```
#  You can also use wget

curl -X PUT https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/sequence/{sequence_name}/{alert_id} \
  -H Content-Type: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Content-Type: application/json,
  X-Auth-Token: string
}

r = requests.put(https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/sequence/{sequence_name}/{alert_id},␣
↪headers = headers)

print(r.json())
```

```
const inputBody = {
  "alert_name": "string",
  "alert_template": 0,
  "description": "string",
  "enabled": true,
  "history_retention": 0,
  "frequency_min": 0,
  "operator": "string",
  "low_threshold_value": 0,
  "medium_threshold_value": 0,
  "high_threshold_value": 0,
```

(次のページに続く)

(前のページからの続き)

```
  "all_alert_enable": true,
  "email_group_id": 1,
  "low_alert_enable": true,
  "low_email_group_id": 1,
  "med_alert_enable": true,
  "med_email_group_id": 1,
  "high_alert_enable": true,
  "high_email_group_id": 1,
  "send_trap": false,
  "snmp_trap_version": 0,
  "low_send_trap": true,
  "med_send_trap": true,
  "high_send_trap": true,
  "cleared_alert_enable": true,
  "params": [],
  "execute_script": true,
  "execute_script_on_clear": true,
  "execute_script_on_pem_server": true,
  "script_code": "string",
  "submit_to_nagios": true,
  "override_default_config": true,
  "send_notification": false,
  "low_webhook_ids": [],
  "med_webhook_ids": [],
  "high_webhook_ids": [],
  "cleared_webhook_ids": []
};
const headers = {
  Content-Type:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/database/
↪{database_name}/schema/{schema_name}/sequence/{sequence_name}/{alert_id},
{
  method: PUT,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

PUT /alert/config/server/{server_id}/database/{database_name}/schema/{schema_name}/sequence/{sequence_name}/{alert_id}

シーケンスレベルの Alert を更新します。

ボディパラメーター

```
{
  "alert_name": "string",
  "alert_template": 0,
  "description": "string",
  "enabled": true,
  "history_retention": 0,
  "frequency_min": 0,
  "operator": "string",
  "low_threshold_value": 0,
  "medium_threshold_value": 0,
  "high_threshold_value": 0,
  "all_alert_enable": true,
  "email_group_id": 1,
  "low_alert_enable": true,
  "low_email_group_id": 1,
  "med_alert_enable": true,
  "med_email_group_id": 1,
  "high_alert_enable": true,
  "high_email_group_id": 1,
  "send_trap": false,
  "snmp_trap_version": 0,
  "low_send_trap": true,
  "med_send_trap": true,
  "high_send_trap": true,
  "cleared_alert_enable": true,
  "params": [],
  "execute_script": true,
  "execute_script_on_clear": true,
  "execute_script_on_pem_server": true,
  "script_code": "string",
  "submit_to_nagios": true,
  "override_default_config": true,
  "send_notification": false,
  "low_webhook_ids": [],
  "med_webhook_ids": [],
  "high_webhook_ids": [],
  "cleared_webhook_ids": []
}
```

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| database_name | path | string | true | データベース名 |
| schema_name | path | string | true | スキーマ名 |
| sequence_name | path | string | true | シークエンス名 |
| alert_id | path | integer | true | アラート ID |
| X-Auth-Token | header | string | true | 承認トークン |
| body | body | *AlertPayload* | true | なし |

### 応答例

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ステータス 200 | None |

### レスポンススキーマ

**注釈**

この操作には認証が必要ありません

## delete_*alert_config_server*{server_id}*database*{database_name}*schema*{schema_name}*sequence*{sequence_na

### コードサンプル

```
#  You can also use wget

curl -X DELETE https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/sequence/{sequence_name}/{alert_id} \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  X-Auth-Token: string
}
```

(次のページに続く)

(前のページからの続き)

```
r = requests.delete(https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/sequence/{sequence_name}/{alert_id},␣
↪headers = headers)

print(r.json())
```

```
const headers = {
  "X-Auth-Token": "string",
};

fetch(
  "https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/database/
↪{database_name}/schema/{schema_name}/sequence/{sequence_name}/{alert_id}",
  {
    method: "DELETE",

    headers: headers,
  },
)
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

`DELETE /alert/config/server/{server_id}/database/{database_name}/schema/{schema_name}/sequence/{sequence_name}/{alert_id}`

`Alert` オブジェクトを削除します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| database_name | path | string | true | データベース名 |
| schema_name | path | string | true | スキーマ名 |
| sequence_name | path | string | true | シークエンス名 |
| alert_id | path | integer | true | アラート ID |
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 削除が成功しました。 | None |

### レスポンススキーマ

> **注釈**
>
> この操作には認証が必要ありません

## get_*alert_config_server*{server_id}*database*{database_name}*schema*{schema_name}*function*{function_name}*ar*

### コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/function/{function_name}/args/
↪{function_arguments}/ \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/function/{function_name}/args/
↪{function_arguments}/, headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch(
```

(次のページに続く)

(前のページからの続き)

```
  "https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/database/
↪{database_name}/schema/{schema_name}/function/{function_name}/args/{function_
↪arguments}/",
  {
    method: "GET",

    headers: headers,
  },
)
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /alert/config/server/{server_id}/database/{database_name}/schema/{schema_name}/function/{function_name}/args/{function_arguments}/

server_id、database_name、schema_name、および function_name ごとにファンクションレベルの Alerts を取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| database_name | path | string | true | データベース名 |
| schema_name | path | string | true | スキーマ名 |
| function_name | path | string | true | ファンクション名 |
| function_arguments | path | string | true | ファンクション引数のコンマ区切りリスト。 |
| X-Auth-Token | header | string | true | 承認トークン |

### 詳細な説明

**function_arguments** ファンクション引数のコンマ区切りリスト。

注ファンクションの引数がない場合は、スペースを入力します

### 応答例

```
[
  {
    "id": 0,
    "alert_name": "string",
```

(次のページに続く)

(前のページからの続き)


```
    "alert_template": 0,
    "description": "string",
    "enabled": true,
    "history_retention": 0,
    "frequency_min": 0,
    "auto_created": true,
    "template_id": "string",
    "params": [],
    "param_names": ["string"],
    "param_types": "string",
    "params_units": ["string"],
    "threshold_unit": "string",
    "operator": "string",
    "low_threshold_value": 0,
    "medium_threshold_value": 0,
    "high_threshold_value": 0,
    "send_email": true,
    "all_alert_enable": true,
    "email_group_id": 0,
    "low_alert_enable": true,
    "low_email_group_id": 0,
    "med_alert_enable": true,
    "med_email_group_id": 0,
    "high_alert_enable": true,
    "high_email_group_id": 0,
    "send_trap": true,
    "snmp_trap_version": 0,
    "low_send_trap": true,
    "med_send_trap": true,
    "high_send_trap": true,
    "cleared_alert_enable": true,
    "execute_script": true,
    "execute_script_on_clear": true,
    "execute_script_on_pem_server": true,
    "script_code": "string",
    "submit_to_nagios": true,
    "override_default_config": true,
    "send_notification": true,
    "low_webhook_ids": [0],
    "med_webhook_ids": [0],
    "high_webhook_ids": [0],
    "cleared_webhook_ids": [0]
  }
]
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | Inline |

### レスポンススキーマ

ステータスコード **200**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| _anonymous_ | *Platform Compatibility* ] | false | none | なし |
| » id | integer | false | none | アラート ID。注このプロパティは、POST 要求には適用されません。 |
| » alert_name | string | false | none | アラートの名前。注 このプロパティは、POST 要求の必須パラメーターです。 |
| » alert_template | integer | false | none | アラートテンプレートの ID。このプロパティは、正の整数値である必要があります。注これはPOST 要求の必須パラメーターです。 |
| » description | string | false | none | なし |
| » enabled | boolean | false | none | このアラートが有効/無効かどうかを判断します。注 このプロパティは、POST 要求の必須パラメーターです。 |
| » history_retention | integer | false | none | このアラートの履歴を保持する日数を指定します。指定できる値は 1 ～99999 です。 注このプロパティは、POST 要求の必須パラメーターです。 |

次のページに続く

表 347 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » frequency_min | integer | false | none | PEM データベースサーバーでアラートを確認する頻度を分単位で定義します。指定できる値は 1〜65534 です。 注このプロパティは、POST 要求の必須パラメーターです。 |
| » auto_created | boolean | false | none | アラートを自動作成するかどうかを指定します。 |
| » template_id | string | false | none | アラートテンプレートを定義します。 |
| » params | *Platform Compatibility* ] | false | none | アラートを確認するためにアラートテンプレートが使用するパラメーター/値のペアのリスト。注アラートテンプレートには、all パラメーターを定義する必要があります。 |
| » paramname | string | false | none | パラメーターの名前。注このプロパティは、アラートテンプレートパラメーターリストに存在する必要があります。例 |
| » paramvalue | string | false | none | 指定されたパラメーターの値 |
| » param_names | [string] | false | none | パラメーターの名前を定義します。 |
| » param_types | string | false | none | パラメーターのタイプを定義します。例 INTEGER/STRING.. |
| » params_units | [string] | false | none | パラメーターに使用される単位を定義します。例 'DAYS'、'True or False'、'Hours'、'Minutes' など |
| » threshold_unit | string | false | none | MB/kB/秒/日/... などの、しきい値の計算に使用される単位 この値は、GUI でのみ使用されます。 |

次のページに続く

表 347 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » operator | string | false | none | しきい値をチェックするための演算子を定義します。有効な文字列は>"および"<"です。注このプロパティは、POST要求の必須パラメーターです。 |
| » low_threshold_value | number | false | none | 低アラートをトリガーするためのしきい値。このプロパティには、有効な整数/浮動小数点値を指定できます。注これはPOST要求の必須パラメーターです。 |
| » medium_threshold_value | number | false | none | 中アラートをトリガーするためのしきい値。このプロパティには、有効な整数/浮動小数点値を指定できます。注これはPOST要求の必須パラメーターです。 |
| » high_threshold_value | number | false | none | 高度なアラートをトリガーするためのしきい値。任意の有効な整数/浮動小数点値を指定できます。注これはPOST要求の必須パラメーターです。 |
| » send_email | boolean | false | none | アラートの状態が変更されたときに電子メールを送信するかどうかを決定します。 |
| » all_alert_enable | boolean | false | none | アラートの状態が変更されたときに、パラメーター 'email_group_id' で指定されたグループにメール通知を送信するかどうかを決定します。 |
| » email_group_id | integer | false | none | アラートの状態が変更されたときに電子メール通知の送信先の電子メールグループを定義します。 |

次のページに続く

表 347 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » low_alert_enable | boolean | false | none | アラートの状態が LOW に設定されているときに、パラメーター 'low_email_group_id' で指定されたグループにメール通知を送信するかどうかを決定します。 |
| » low_email_group_id | integer | false | none | アラートの状態が LOW に変更されたときに電子メール通知の送信先の電子メールグループを定義します。 |
| » med_alert_enable | boolean | false | none | アラートの状態が MEDIUM に設定されている場合、パラメーター 'medium_email_group_id' で指定されたグループにメール通知を送信するかどうかを決定します。 |
| » med_email_group_id | integer | false | none | アラートの状態が MEDIUM に変更されたときにメール通知の送信先のメールグループを定義します。 |
| » high_alert_enable | boolean | false | none | アラートの状態が HIGH に設定されているときに、パラメーター 'high_email_group_id' で指定されたグループに電子メール通知を送信するかどうかを決定します。 |
| » high_email_group_id | integer | false | none | アラートの状態が HIGH に変更されたときに電子メール通知の送信先の電子メールグループを定義します。 |
| » send_trap | boolean | false | none | アラートの状態が変更されたときに SNMP トラップ通知を送信するかどうかを決定します。 |

次のページに続く

表 347 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » snmp_trap_version | integer | false | none | SNMP トラップバージョンを定義します。サポートされているSNMP トラップバージョンは 1、2、または 3 です |
| » low_send_trap | boolean | false | none | アラートの状態が LOW に変更されたときに SNMP トラップ通知を送信するかどうかを決定します。 |
| » med_send_trap | boolean | false | none | アラートの状態が MEDIUM に変更されたときに SNMP トラップ通知を送信するかどうかを決定します。 |
| » high_send_trap | boolean | false | none | アラートの状態が HIGH に変わったときに SNMP トラップ通知を送信するかどうかを決定します。 |
| » cleared_alert_enable | boolean | false | none | ' クリアアラート' の通知をミュートするかどうかを決定します。 |
| » execute_script | boolean | false | none | アラートの状態が LOW/MEDIUM/HIGH に変更された場合に、提供されたスクリプトを実行するかどうかを決定します。 |
| » execute_script_on_clear | boolean | false | none | アラートの状態がクリアされた場合に、提供されたスクリプトを実行するかどうかを決定します。 |
| » execute_script_on_pem_serve | boolean | false | none | 提供されたスクリプトを PEM ホストまたはエージェントがオブジェクトを監視しているホストで実行するかどうかを決定します。 |

次のページに続く

表 347 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » script_code | string | false | none | 実行するシェルまたはバッチスクリプト。詳細については、ドキュメントをご覧ください。 |
| » submit_to_nagios | boolean | false | none | アラートの状態が変更されたときに nagios に通知を送信するかどうかを決定します。 |
| » override_default_config | boolean | false | none | デフォルトの Webhook 構成をオーバーライドするかどうかを決定します。 |
| » send_notification | boolean | false | none | Webhook 送信通知を有効にするかどうかを決定します。 |
| » low_webhook_ids | [integer] | false | none | 低アラートが構成されている Webhook ID |
| » med_webhook_ids | [integer] | false | none | 中アラートが構成されている Webhook ID |
| » high_webhook_ids | [integer] | false | none | 高アラートが構成されている Webhook ID |
| » cleared_webhook_ids | [integer] | false | none | クリアされたアラートが構成されている Webhook ID |

### 列挙値

| Property | Value |
|---|---|
| paramname | Exclude tables smaller than |
| paramname | mount point |
| paramname | Host IP Address |
| paramname | Time interval to monitor |
| paramname | Time since idle |
| paramname | Per CPU utilization (K%) |

**注釈**

この操作には認証が必要ありません

### post_*alert_config_server*{server_id}*database*{database_name}*schema*{schema_name}*function*{function_name}a

コードサンプル

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/function/{function_name}/args/
↪{function_arguments}/ \
  -H Content-Type: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Content-Type: application/json,
  X-Auth-Token: string
}

r = requests.post(https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/function/{function_name}/args/
↪{function_arguments}/, headers = headers)

print(r.json())
```

```
const inputBody = {
  "alert_name": "string",
  "alert_template": 0,
  "description": "string",
  "enabled": true,
  "history_retention": 0,
  "frequency_min": 0,
  "operator": "string",
  "low_threshold_value": 0,
  "medium_threshold_value": 0,
  "high_threshold_value": 0,
  "all_alert_enable": true,
  "email_group_id": 1,
  "low_alert_enable": true,
  "low_email_group_id": 1,
  "med_alert_enable": true,
  "med_email_group_id": 1,
  "high_alert_enable": true,
  "high_email_group_id": 1,
  "send_trap": false,
  "snmp_trap_version": 0,
  "low_send_trap": true,
```

(次のページに続く)

(前のページからの続き)

```
  "med_send_trap": true,
  "high_send_trap": true,
  "cleared_alert_enable": true,
  "params": [],
  "execute_script": true,
  "execute_script_on_clear": true,
  "execute_script_on_pem_server": true,
  "script_code": "string",
  "submit_to_nagios": true,
  "override_default_config": true,
  "send_notification": false,
  "low_webhook_ids": [],
  "med_webhook_ids": [],
  "high_webhook_ids": [],
  "cleared_webhook_ids": []
};
const headers = {
  Content-Type:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/database/
↪{database_name}/schema/{schema_name}/function/{function_name}/args/{function_
↪arguments}/,
{
  method: POST,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

POST /alert/config/server/{server_id}/database/{database_name}/schema/{schema_name}/function/{function_name}/args/{function_arguments}/

ファンクションレベルの Alert を作成します。

**ボディパラメーター**

```
{
  "alert_name": "string",
  "alert_template": 0,
  "description": "string",
```

(次のページに続く)

(前のページからの続き)


```
  "enabled": true,
  "history_retention": 0,
  "frequency_min": 0,
  "operator": "string",
  "low_threshold_value": 0,
  "medium_threshold_value": 0,
  "high_threshold_value": 0,
  "all_alert_enable": true,
  "email_group_id": 1,
  "low_alert_enable": true,
  "low_email_group_id": 1,
  "med_alert_enable": true,
  "med_email_group_id": 1,
  "high_alert_enable": true,
  "high_email_group_id": 1,
  "send_trap": false,
  "snmp_trap_version": 0,
  "low_send_trap": true,
  "med_send_trap": true,
  "high_send_trap": true,
  "cleared_alert_enable": true,
  "params": [],
  "execute_script": true,
  "execute_script_on_clear": true,
  "execute_script_on_pem_server": true,
  "script_code": "string",
  "submit_to_nagios": true,
  "override_default_config": true,
  "send_notification": false,
  "low_webhook_ids": [],
  "med_webhook_ids": [],
  "high_webhook_ids": [],
  "cleared_webhook_ids": []
}
```


### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| database_name | path | string | true | データベース名 |
| schema_name | path | string | true | スキーマ名 |
| function_name | path | string | true | ファンクション名 |

次のページに続く

表 349 – 前のページからの続き

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| function_arguments | path | string | true | ファンクション引数のコンマ区切りリスト。 |
| X-Auth-Token | header | string | true | 承認トークン |
| body | body | *AlertPayload* | true | なし |

### 詳細な説明

**function_arguments** ファンクション引数のコンマ区切りリスト。

注ファンクションの引数がない場合は、スペースを入力します

### 応答例

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ステータス 200 | None |

### レスポンススキーマ

**注釈**

この操作には認証が必要ありません

## get_*alert_config_server*{server_id}*database*{database_name}*schema*{schema_name}*function*{function_name}*ar*

### コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/function/{function_name}/args/
↪{function_arguments}/{alert_id} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
```

(次のページに続く)

(前のページからの続き)

```
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/function/{function_name}/args/
↪{function_arguments}/{alert_id}, headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch(
  "https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/database/
↪{database_name}/schema/{schema_name}/function/{function_name}/args/{function_
↪arguments}/{alert_id}",
  {
    method: "GET",

    headers: headers,
  },
)
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

`GET /alert/config/server/{server_id}/database/{database_name}/schema/{schema_name}/function/{function_name}/args/{function_arguments}/{alert_id}`

server_id、database_name、schema_name、function_name、および alert_id ごとにファンクションレベルの Alerts を取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| database_name | path | string | true | データベース名 |
| schema_name | path | string | true | スキーマ名 |

次のページに続く

表 351 – 前のページからの続き

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| function_name | path | string | true | ファンクション名 |
| function_arguments | path | string | true | ファンクション引数のコンマ区切りリスト。 |
| alert_id | path | integer | true | アラート ID |
| X-Auth-Token | header | string | true | 承認トークン |

#### 詳細な説明

**function_arguments** ファンクション引数のコンマ区切りリスト。

注ファンクションの引数がない場合は、スペースを入力します

#### 応答例

```
{
  "id": 0,
  "alert_name": "string",
  "alert_template": 0,
  "description": "string",
  "enabled": true,
  "history_retention": 0,
  "frequency_min": 0,
  "auto_created": true,
  "template_id": "string",
  "params": [],
  "param_names": ["string"],
  "param_types": "string",
  "params_units": ["string"],
  "threshold_unit": "string",
  "operator": "string",
  "low_threshold_value": 0,
  "medium_threshold_value": 0,
  "high_threshold_value": 0,
  "send_email": true,
  "all_alert_enable": true,
  "email_group_id": 0,
  "low_alert_enable": true,
  "low_email_group_id": 0,
  "med_alert_enable": true,
  "med_email_group_id": 0,
  "high_alert_enable": true,
  "high_email_group_id": 0,
  "send_trap": true,
  "snmp_trap_version": 0,
```

(次のページに続く)

(前のページからの続き)

```
  "low_send_trap": true,
  "med_send_trap": true,
  "high_send_trap": true,
  "cleared_alert_enable": true,
  "execute_script": true,
  "execute_script_on_clear": true,
  "execute_script_on_pem_server": true,
  "script_code": "string",
  "submit_to_nagios": true,
  "override_default_config": true,
  "send_notification": true,
  "low_webhook_ids": [0],
  "med_webhook_ids": [0],
  "high_webhook_ids": [0],
  "cleared_webhook_ids": [0]
}
```

応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | アラートスレッドについて |

**注釈**

この操作には認証が必要ありません

### put_*alert_config_server*{server_id}*database*{database_name}*schema*{schema_name}*function*{function_name}*ar*

コードサンプル

```
#  You can also use wget

curl -X PUT https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/function/{function_name}/args/
↪{function_arguments}/{alert_id} \
  -H Content-Type: application/json \
  -H X-Auth-Token: string
```

```
import requests
```

(次のページに続く)

(前のページからの続き)

```
headers = {
  Content-Type: application/json,
  X-Auth-Token: string
}

r = requests.put(https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/function/{function_name}/args/
↪{function_arguments}/{alert_id}, headers = headers)

print(r.json())
```

```
const inputBody = {
  "alert_name": "string",
  "alert_template": 0,
  "description": "string",
  "enabled": true,
  "history_retention": 0,
  "frequency_min": 0,
  "operator": "string",
  "low_threshold_value": 0,
  "medium_threshold_value": 0,
  "high_threshold_value": 0,
  "all_alert_enable": true,
  "email_group_id": 1,
  "low_alert_enable": true,
  "low_email_group_id": 1,
  "med_alert_enable": true,
  "med_email_group_id": 1,
  "high_alert_enable": true,
  "high_email_group_id": 1,
  "send_trap": false,
  "snmp_trap_version": 0,
  "low_send_trap": true,
  "med_send_trap": true,
  "high_send_trap": true,
  "cleared_alert_enable": true,
  "params": [],
  "execute_script": true,
  "execute_script_on_clear": true,
  "execute_script_on_pem_server": true,
  "script_code": "string",
  "submit_to_nagios": true,
  "override_default_config": true,
  "send_notification": false,
```

(次のページに続く)

(前のページからの続き)

```
  "low_webhook_ids": [],
  "med_webhook_ids": [],
  "high_webhook_ids": [],
  "cleared_webhook_ids": []
};
const headers = {
  Content-Type:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/database/
↪{database_name}/schema/{schema_name}/function/{function_name}/args/{function_
↪arguments}/{alert_id},
{
  method: PUT,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

PUT /alert/config/server/{server_id}/database/{database_name}/schema/{schema_name}/function/{function_name}/args/{function_arguments}/{alert_id}

ファンクションレベルの Alert を更新します。

**ボディパラメーター**

```
{
  "alert_name": "string",
  "alert_template": 0,
  "description": "string",
  "enabled": true,
  "history_retention": 0,
  "frequency_min": 0,
  "operator": "string",
  "low_threshold_value": 0,
  "medium_threshold_value": 0,
  "high_threshold_value": 0,
  "all_alert_enable": true,
  "email_group_id": 1,
  "low_alert_enable": true,
  "low_email_group_id": 1,
```

(次のページに続く)

(前のページからの続き)


```
  "med_alert_enable": true,
  "med_email_group_id": 1,
  "high_alert_enable": true,
  "high_email_group_id": 1,
  "send_trap": false,
  "snmp_trap_version": 0,
  "low_send_trap": true,
  "med_send_trap": true,
  "high_send_trap": true,
  "cleared_alert_enable": true,
  "params": [],
  "execute_script": true,
  "execute_script_on_clear": true,
  "execute_script_on_pem_server": true,
  "script_code": "string",
  "submit_to_nagios": true,
  "override_default_config": true,
  "send_notification": false,
  "low_webhook_ids": [],
  "med_webhook_ids": [],
  "high_webhook_ids": [],
  "cleared_webhook_ids": []
}
```


### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| database_name | path | string | true | データベース名 |
| schema_name | path | string | true | スキーマ名 |
| function_name | path | string | true | ファンクション名 |
| function_arguments | path | string | true | ファンクション引数のコンマ区切りリスト。 |
| alert_id | path | integer | true | アラート ID |
| X-Auth-Token | header | string | true | 承認トークン |
| body | body | *AlertPayload* | true | なし |

### 詳細な説明

**function_arguments** ファンクション引数のコンマ区切りリスト。

注ファンクションの引数がない場合は、スペースを入力します

### 応答例

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ステータス 200 | None |

### レスポンススキーマ

> **注釈**
>
> この操作には認証が必要ありません

## delete_*alert_config_server*{server_id}*database*{database_name}*schema*{schema_name}*function*{function_name

### コードサンプル

```
#  You can also use wget

curl -X DELETE https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/function/{function_name}/args/
↪{function_arguments}/{alert_id} \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  X-Auth-Token: string
}

r = requests.delete(https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/function/{function_name}/args/
↪{function_arguments}/{alert_id}, headers = headers)

print(r.json())
```

```
const headers = {
  "X-Auth-Token": "string",
};

fetch(
  "https://PEM-SERVER-IP/pem/api/v17/alert/config/server/{server_id}/database/
↪{database_name}/schema/{schema_name}/function/{function_name}/args/{function_
↪arguments}/{alert_id}",
```

(次のページに続く)

(前のページからの続き)

```
  {
    method: "DELETE",

    headers: headers,
  },
)
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

`DELETE /alert/config/server/{server_id}/database/{database_name}/schema/{schema_name}/function/{function_name}/args/{function_arguments}/{alert_id}`

`Alert` オブジェクトを削除します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| database_name | path | string | true | データベース名 |
| schema_name | path | string | true | スキーマ名 |
| function_name | path | string | true | ファンクション名 |
| function_arguments | path | string | true | ファンクション引数のコンマ区切りリスト。 |
| alert_id | path | integer | true | アラート ID |
| X-Auth-Token | header | string | true | 承認トークン |

### 詳細な説明

**function_arguments** ファンクション引数のコンマ区切りリスト。

注ファンクションの引数がない場合は、スペースを入力します

**応答例**

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 削除が成功しました。 | None |

レスポンススキーマ

> **注釈**
>
> この操作には認証が必要ありません

### Copy Alert

#### post_*alert_copy_agent*{agent_id}

コードサンプル

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v17/alert/copy/agent/{agent_id} \
  -H Content-Type: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Content-Type: application/json,
  X-Auth-Token: string
}

r = requests.post(https://PEM-SERVER-IP/pem/api/v17/alert/copy/agent/{agent_id},␣
↪headers = headers)

print(r.json())
```

```
const inputBody = [
  {
    "type": "string",
    "agent_id": 0,
    "existing_alert_options": "I"
  }
];
const headers = {
  Content-Type:application/json,
  X-Auth-Token:string
```

(次のページに続く)

(前のページからの続き)

```
};

fetch(https://PEM-SERVER-IP/pem/api/v17/alert/copy/agent/{agent_id},
{
  method: POST,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

POST /alert/copy/agent/{agent_id}

エージェントレベルのアラートを別のエージェントにコピーします。

**ボディパラメーター**

```
[
  {
    "type": "string",
    "agent_id": 0,
    "existing_alert_options": "I"
  }
]
```

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| agent_id | path | integer | true | エージェント ID |
| X-Auth-Token | header | string | true | 承認トークン |
| body | body | *CopyAlertAgentData* | true | なし |

### 詳細な説明

**body** 注

- **type** 値は’agent’である必要があります

**応答例**

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ステータス 200 | None |

### レスポンススキーマ

> **注釈**
>
> この操作には認証が必要ありません

### post_alert_copy_server{server_id}

#### コードサンプル

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v17/alert/copy/server/{server_id} \
  -H Content-Type: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Content-Type: application/json,
  X-Auth-Token: string
}

r = requests.post(https://PEM-SERVER-IP/pem/api/v17/alert/copy/server/{server_id},␣
↪headers = headers)

print(r.json())
```

```
const inputBody = [
  {
    "type": "string",
    "server_id": 0,
    "existing_alert_options": "I"
  }
];
const headers = {
  Content-Type:application/json,
```

(次のページに続く)

(前のページからの続き)

```
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v17/alert/copy/server/{server_id},
{
  method: POST,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

POST /alert/copy/server/{server_id}

サーバーレベルのアラートを別のサーバーにコピーします。

**ボディパラメーター**

```
[
  {
    "type": "string",
    "server_id": 0,
    "existing_alert_options": "I"
  }
]
```

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| X-Auth-Token | header | string | true | 承認トークン |
| body | body | *CopyAlert-ServerData* | true | なし |

### 詳細な説明

**body** 注

- **type** 値は’server’である必要があります

**応答例**

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ステータス 200 | None |

### レスポンススキーマ

> **注釈**
>
> この操作には認証が必要ありません

### post_alert_copy_server{server_id}database{database_name}

#### コードサンプル

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v17/alert/copy/server/{server_id}/database/
↪{database_name} \
  -H Content-Type: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Content-Type: application/json,
  X-Auth-Token: string
}

r = requests.post(https://PEM-SERVER-IP/pem/api/v17/alert/copy/server/{server_id}/
↪database/{database_name}, headers = headers)

print(r.json())
```

```
const inputBody = [
  {
    "type": "string",
    "server_id": 0,
    "database_name": "string",
    "existing_alert_options": "I"
  }
];
```

(次のページに続く)

(前のページからの続き)

```
const headers = {
  Content-Type:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v17/alert/copy/server/{server_id}/database/
↪{database_name},
{
  method: POST,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

POST /alert/copy/server/{server_id}/database/{database_name}

データベースレベルのアラートを別のデータベースにコピーします。

**ボディパラメーター**

```
[
  {
    "type": "string",
    "server_id": 0,
    "database_name": "string",
    "existing_alert_options": "I"
  }
]
```

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| database_name | path | string | true | データベース名 |
| X-Auth-Token | header | string | true | 承認トークン |
| body | body | *CopyAlert-Database-Data* | true | なし |

### 詳細な説明

**body** 注

- **type** 値は’database’である必要があります

**応答例**

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ステータス 200 | None |

### レスポンススキーマ

> **注釈**
>
> この操作には認証が必要ありません

### post_alert_copy_server{server_id}database{database_name}schema{schema_name}

**コードサンプル**

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v17/alert/copy/server/{server_id}/database/
↪{database_name}/schema/{schema_name} \
  -H Content-Type: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Content-Type: application/json,
  X-Auth-Token: string
}

r = requests.post(https://PEM-SERVER-IP/pem/api/v17/alert/copy/server/{server_id}/
↪database/{database_name}/schema/{schema_name}, headers = headers)

print(r.json())
```

```
const inputBody = [
  {
    "type": "string",
    "server_id": 0,
    "database_name": "string",
    "schema_name": "string",
    "existing_alert_options": "I"
  }
];
const headers = {
  Content-Type:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v17/alert/copy/server/{server_id}/database/
↪{database_name}/schema/{schema_name},
{
  method: POST,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

POST /alert/copy/server/{server_id}/database/{database_name}/schema/{schema_name}

スキーマレベルのアラートを別のスキーマにコピーします。

**ボディパラメーター**

```
[
  {
    "type": "string",
    "server_id": 0,
    "database_name": "string",
    "schema_name": "string",
    "existing_alert_options": "I"
  }
]
```

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| database_name | path | string | true | データベース名 |
| schema_name | path | string | true | スキーマ名 |
| X-Auth-Token | header | string | true | 承認トークン |
| body | body | *Copy-AlertSchema-Data* | true | なし |

### 詳細な説明

**body** 注

- **type** 値は’schema’である必要があります

**応答例**

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ステータス 200 | None |

### レスポンススキーマ

**注釈**

この操作には認証が必要ありません

## post_*alert_copy_server*{server_id}*database*{database_name}*schema*{schema_name}*table*{table_name}

**コードサンプル**

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v17/alert/copy/server/{server_id}/database/
↪{database_name}/schema/{schema_name}/table/{table_name} \
  -H Content-Type: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Content-Type: application/json,
  X-Auth-Token: string
}

r = requests.post(https://PEM-SERVER-IP/pem/api/v17/alert/copy/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/table/{table_name}, headers = headers)

print(r.json())
```

```
const inputBody = [
  {
    "type": "string",
    "server_id": 0,
    "database_name": "string",
    "schema_name": "string",
    "table_name": "string",
    "existing_alert_options": "I"
  }
];
const headers = {
  Content-Type:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v17/alert/copy/server/{server_id}/database/
↪{database_name}/schema/{schema_name}/table/{table_name},
{
  method: POST,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

POST /alert/copy/server/{server_id}/database/{database_name}/schema/{schema_name}/table/{table_name}

テーブルレベルのアラートを別のテーブルにコピーします。

**ボディパラメーター**

```
[
  {
    "type": "string",
    "server_id": 0,
    "database_name": "string",
    "schema_name": "string",
    "table_name": "string",
    "existing_alert_options": "I"
  }
]
```

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| database_name | path | string | true | データベース名 |
| schema_name | path | string | true | スキーマ名 |
| table_name | path | string | true | テーブル名 |
| X-Auth-Token | header | string | true | 承認トークン |
| body | body | *CopyAlert-TableData* | true | なし |

### 詳細な説明

**body** 注

- **type** 値は’table’である必要があります

**応答例**

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ステータス 200 | None |

### レスポンススキーマ

**注釈**

この操作には認証が必要ありません

### post_*alert_copy_server*{server_id}*database*{database_name}*schema*{schema_name}*index*{index_name}

コードサンプル

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v17/alert/copy/server/{server_id}/database/
↪{database_name}/schema/{schema_name}/index/{index_name} \
  -H Content-Type: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Content-Type: application/json,
  X-Auth-Token: string
}

r = requests.post(https://PEM-SERVER-IP/pem/api/v17/alert/copy/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/index/{index_name}, headers = headers)

print(r.json())
```

```
const inputBody = [
  {
    "type": "string",
    "server_id": 0,
    "database_name": "string",
    "schema_name": "string",
    "index_name": "string",
    "existing_alert_options": "I"
  }
];
const headers = {
  Content-Type:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v17/alert/copy/server/{server_id}/database/
↪{database_name}/schema/{schema_name}/index/{index_name},
{
  method: POST,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
```

(次のページに続く)

(前のページからの続き)

```
}).then(function(body) {
    console.log(body);
});
```

POST /alert/copy/server/{server_id}/database/{database_name}/schema/{schema_name}/index/{index_name}

インデックスレベルのアラートを別のインデックスにコピーします。

**ボディパラメーター**

```
[
  {
    "type": "string",
    "server_id": 0,
    "database_name": "string",
    "schema_name": "string",
    "index_name": "string",
    "existing_alert_options": "I"
  }
]
```

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| database_name | path | string | true | データベース名 |
| schema_name | path | string | true | スキーマ名 |
| index_name | path | string | true | インデックス名 |
| X-Auth-Token | header | string | true | 承認トークン |
| body | body | *Copy-AlertIndex-Data* | true | なし |

### 詳細な説明

**body** 注

- **type** 値は’index’である必要があります

**応答例**

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ステータス 200 | None |

### レスポンススキーマ

> **注釈**
>
> この操作には認証が必要ありません

## post_alert_copy_server{server_id}database{database_name}schema{schema_name}function{function_name}ar

### コードサンプル

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v17/alert/copy/server/{server_id}/database/
↪{database_name}/schema/{schema_name}/function/{function_name}/args/{function_
↪arguments} \
  -H Content-Type: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Content-Type: application/json,
  X-Auth-Token: string
}

r = requests.post(https://PEM-SERVER-IP/pem/api/v17/alert/copy/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/function/{function_name}/args/
↪{function_arguments}, headers = headers)

print(r.json())
```

```
const inputBody = [
  {
    "type": "string",
    "server_id": 0,
    "database_name": "string",
    "schema_name": "string",
```

(次のページに続く)

(前のページからの続き)

```
    "function_name": "string",
    "function_arguments": "string",
    "existing_alert_options": "I"
  }
];
const headers = {
  Content-Type:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v17/alert/copy/server/{server_id}/database/
↪{database_name}/schema/{schema_name}/function/{function_name}/args/{function_
↪arguments},
{
  method: POST,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

POST /alert/copy/server/{server_id}/database/{database_name}/schema/{schema_name}/function/{function_name}/args/{function_arguments}

ファンクションレベルのアラートを別のファンクションにコピーします。

**ボディパラメーター**

```
[
  {
    "type": "string",
    "server_id": 0,
    "database_name": "string",
    "schema_name": "string",
    "function_name": "string",
    "function_arguments": "string",
    "existing_alert_options": "I"
  }
]
```

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| database_name | path | string | true | データベース名 |
| schema_name | path | string | true | スキーマ名 |
| function_name | path | string | true | ファンクション名 |
| function_arguments | path | string | true | ファンクション引数のコンマ区切りリスト。 |
| X-Auth-Token | header | string | true | 承認トークン |
| body | body | *CopyAlert-Function-Data* | true | なし |

### 詳細な説明

**function_arguments** ファンクション引数のコンマ区切りリスト。

注ファンクションの引数がない場合は、スペースを入力します

**body** 注

- **type** 値は’function’である必要があります

**応答例**

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ステータス 200 | None |

### レスポンススキーマ

> **注釈**
>
> この操作には認証が必要ありません

**post_*alert_copy_server*{server_id}*database*{database_name}*schema*{schema_name}*sequence*{sequence_name**

**コードサンプル**

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v17/alert/copy/server/{server_id}/database/
↪{database_name}/schema/{schema_name}/sequence/{sequence_name} \
  -H Content-Type: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Content-Type: application/json,
  X-Auth-Token: string
}

r = requests.post(https://PEM-SERVER-IP/pem/api/v17/alert/copy/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/sequence/{sequence_name}, headers =␣
↪headers)

print(r.json())
```

```
const inputBody = [
  {
    "type": "string",
    "server_id": 0,
    "database_name": "string",
    "schema_name": "string",
    "sequence_name": "string",
    "existing_alert_options": "I"
  }
];
const headers = {
  Content-Type:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v17/alert/copy/server/{server_id}/database/
↪{database_name}/schema/{schema_name}/sequence/{sequence_name},
{
  method: POST,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
```

(次のページに続く)

(前のページからの続き)

```
});
```

`POST /alert/copy/server/{server_id}/database/{database_name}/schema/{schema_name}/sequence/{sequence_name}`

シーケンスレベルのアラートを別のシーケンスにコピーします。

**ボディパラメーター**

```
[
  {
    "type": "string",
    "server_id": 0,
    "database_name": "string",
    "schema_name": "string",
    "sequence_name": "string",
    "existing_alert_options": "I"
  }
]
```

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| database_name | path | string | true | データベース名 |
| schema_name | path | string | true | スキーマ名 |
| sequence_name | path | string | true | シークエンス名 |
| X-Auth-Token | header | string | true | 承認トークン |
| body | body | *CopyAlert-Sequence-Data* | true | なし |

### 詳細な説明

**body** 注

- **type** 値は’sequence’である必要があります

**応答例**

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ステータス 200 | None |

レスポンススキーマ

> **注釈**
>
> この操作には認証が必要ありません

### Alert template

#### get_*alert_template*

コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/alert/template/ \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/alert/template/, headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch("https://PEM-SERVER-IP/pem/api/v17/alert/template/", {
  method: "GET",

  headers: headers,
})
  .then(function (res) {
```

(次のページに続く)

(前のページからの続き)

```
  return res.json();
})
.then(function (body) {
  console.log(body);
});
```

GET /alert/template/

アラートテンプレートを取得します。 name 完全一致または name_contains 部分文字列検索クエリパラメーターを介してオプションの名前ベースのフィルタリングをサポートします。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| X-Auth-Token | header | string | true | 承認トークン |
| name | query | string | false | 名前の完全な一致大文字と小文字を区別してアラートテンプレートをフィルタリングします。name_contains と相互排他的。 |
| name_contains | query | string | false | 部分文字列の一致大文字と小文字を区別してアラートテンプレートをフィルタリングします。 name と相互排他的。 |

### 応答例

```
[
  {
    "name": "string",
    "description": "string",
    "object_type": 0,
    "param_names": ["string"],
    "param_types": "string",
    "params_units": ["string"],
    "sql": "string",
    "applicable_on_server": "ALL",
    "default_check_frequency": 0,
    "default_history_retention": 0,
    "probe_dependency_list": ["string"],
    "threshold_unit": "string",
    "is_system_template": false,
    "is_auto_create": true,
    "operator": ">",
    "low_threshold_value": 0,
```

(次のページに続く)

(前のページからの続き)

```
    "medium_threshold_value": 0,
    "high_threshold_value": 0,
    "info_sql": "string"
  }
]
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | Inline |
| 400 | Bad Request | 相互排他性違反 `name`または類似の完全一致フィルタと`name_contains`または類似の部分文字列フィルタが両方とも同じ要求で指定されました。フィルタリングは、ペアのうちの 1 つのみを受け入れます。 | Inline |

### レスポンススキーマ

ステータスコード **200**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| _anonymous_ | *Platform Compatibility* ] | false | none | なし |
| » name | string | false | none | アラートテンプレートの名前。注 このプロパティは、POST 要求の必須パラメーターです。 |
| » description | string | false | none | アラートテンプレートの説明。注 このプロパティは、POST 要求の必須パラメーターです。 |

次のページに続く

表 375 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » object_type | integer | false | none | このアラートテンプレートを適用するオブジェクトのタイプを定義します。<ul><li>システム全体のアラートテンプレートは50です。</li><li>エージェントに適用されるアラートテンプレートは100です。</li> <li>管理対象サーバーに適用されるアラートテンプレートは200です。</li><li>特定の監視対象サーバーのデータベースに適用されるアラートテンプレートは300です。</li><li>管理対象サーバーの特定のデータベースのスキーマに適用されるアラートテンプレートは400です。</li><li>特定のスキーマのテーブルに適用されるアラートテンプレート</li><li>監視対象サーバー上のデータベースの特定のスキーマのインデックスに適用されるアラートテンプレートは600です。>.</li><li>監視対象サーバー上のデータベースの特定のスキーマのシーケンスに適用できるアラートテンプレートは700です。</li><li>に適用できるアラートテンプレート監視対象サーバー上のデータベースの特定のスキーマのファンクションは800で |

表 375 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » param_names | [string] | false | none | パラメーターの名前を定義します。 |
| » param_types | string | false | none | パラメーターのタイプを定義します。例 INTEGER/STRING.. |
| » params_units | [string] | false | none | パラメーターに使用される単位を定義します。例 'DAYS'、'True or False'、'Hours'、'Minutes' など |
| » sql | string | false | none | しきい値に到達したことを示す数値を取得するための SQL クエリー。パラメーター化される場合があります。注 このプロパティは、POST 要求の必須パラメーターです。 |
| » applicable_on_server | string | false | none | テンプレートを適用する監視対象サーバーの種類を指定します。使用可能な値は、ALL、POSTGRES_SERVER、および ADVANCED_SERVER です。注 このプロパティは、POST 要求の必須パラメーターです。 |
| » default_check_frequency | integer | false | none | このテンプレートで定義されたアラートをチェックする頻度を分単位で定義します。注 このプロパティは、POST 要求の必須パラメーターです。 |
| » default_history_retention | integer | false | none | このタイプのアラートの履歴保持期間を日単位で定義します。注 このプロパティは、POST 要求の必須パラメーターです。 |
| » probe_dependency_list | [string] | false | none | アラートテンプレートが依存するプローブのリストを定義します。 |

次のページに続く

表 375 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » threshold_unit | string | false | none | MB/kB/秒/日/... などの、しきい値の計算に使用される単位 この値は、GUI でのみ使用されます。 |
| » is_system_template | boolean | false | none | 指定されたテンプレートがシステム定義であるかユーザー定義であるかを判断します。注このプロパティは、POST/PUT 要求には適用されません。 |
| » is_auto_create | boolean | false | none | エージェントまたはサーバーが PEM に追加されたらすぐにアラートを作成するかどうかを決定します。注 エージェントおよびサーバーレベル、つまり object_type 100 および 200 でのみサポートされています。 |
| » operator | string | false | none | しきい値を比較する演算子。エージェントおよびサーバーレベル、つまり object_type 100 および 200 でのみサポートされています。 |
| » low_threshold_value | number | false | none | 低しきい値。エージェントおよびサーバーレベル、つまり object_type 100 および 200 でのみサポートされています。 |
| » medium_threshold_value | number | false | none | 中しきい値。エージェントおよびサーバーレベル、つまり object_type 100 および 200 でのみサポートされています。 |
| » high_threshold_value | number | false | none | 高しきい値。エージェントおよびサーバーレベル、つまり object_type 100 および 200 でのみサポートされています。 |

次のページに続く

表 375 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » info_sql | string | false | none | ダッシュボードのアラートに関する詳細情報を提供する SQL クエリーを指定します。 |

### 列挙値

| Property | Value |
|---|---|
| applicable_on_server | ALL |
| applicable_on_server | POSTGRES_SERVER |
| applicable_on_server | ADVANCED_SERVER |
| operator | > |
| operator | < |

ステータスコード **400**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » success | integer | false | none | なし |
| » errormsg | string | false | none | なし |

**注釈**

この操作には認証が必要ありません

### post_alert_template

#### コードサンプル

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v17/alert/template/ \
  -H Content-Type: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Content-Type: application/json,
  X-Auth-Token: string
}
```

(次のページに続く)

(前のページからの続き)

```
r = requests.post(https://PEM-SERVER-IP/pem/api/v17/alert/template/, headers =␣
↪headers)

print(r.json())
```

```
const inputBody = {
  "name": "string",
  "description": "string",
  "object_type": 0,
  "params": [
    {
      "param_types": "STRING",
      "param_units": "KK",
      "param_names": "param_1"
    }
  ],
  "sql": "string",
  "applicable_on_server": "ALL",
  "default_check_frequency": 0,
  "default_history_retention": 0,
  "probe_dependency_list": [
    {
      "internal_name": "probe_name"
    }
  ],
  "threshold_unit": "string",
  "is_system_template": false,
  "is_auto_create": true,
  "operator": ">",
  "low_threshold_value": 0,
  "medium_threshold_value": 0,
  "high_threshold_value": 0,
  "info_sql": "string"
};
const headers = {
  Content-Type:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v17/alert/template/,
{
  method: POST,
  body: inputBody,
  headers: headers
```

(次のページに続く)

(前のページからの続き)

```
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

POST /alert/template/

アラートテンプレートを作成します。

**ボディパラメーター**

```
{
  "name": "string",
  "description": "string",
  "object_type": 0,
  "params": [
    {
      "param_types": "STRING",
      "param_units": "KK",
      "param_names": "param_1"
    }
  ],
  "sql": "string",
  "applicable_on_server": "ALL",
  "default_check_frequency": 0,
  "default_history_retention": 0,
  "probe_dependency_list": [
    {
      "internal_name": "probe_name"
    }
  ],
  "threshold_unit": "string",
  "is_system_template": false,
  "is_auto_create": true,
  "operator": ">",
  "low_threshold_value": 0,
  "medium_threshold_value": 0,
  "high_threshold_value": 0,
  "info_sql": "string"
}
```

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| X-Auth-Token | header | string | true | 承認トークン |
| body | body | *AlertTemplatePost* | true | なし |

#### 応答例

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ステータス 200 | None |

### レスポンススキーマ

**注釈**

この操作には認証が必要ありません

## get_*alert_template*{id}

#### コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/alert/template/{id} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/alert/template/{id}, headers =␣
↪headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch("https://PEM-SERVER-IP/pem/api/v17/alert/template/{id}", {
  method: "GET",

  headers: headers,
})
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /alert/template/{id}

ID で識別されるアラートテンプレートを取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| id | path | integer | true | アラートテンプレート ID |
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例

```
{
  "name": "string",
  "description": "string",
  "object_type": 0,
  "param_names": ["string"],
  "param_types": "string",
  "params_units": ["string"],
  "sql": "string",
  "applicable_on_server": "ALL",
  "default_check_frequency": 0,
  "default_history_retention": 0,
  "probe_dependency_list": ["string"],
  "threshold_unit": "string",
  "is_system_template": false,
  "is_auto_create": true,
  "operator": ">",
```

(次のページに続く)

(前のページからの続き)

```
  "low_threshold_value": 0,
  "medium_threshold_value": 0,
  "high_threshold_value": 0,
  "info_sql": "string"
}
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | *AlertTemplateParams* |

**注釈**

この操作には認証が必要ありません

### put_*alert_template*{id}

### コードサンプル

```
#  You can also use wget

curl -X PUT https://PEM-SERVER-IP/pem/api/v17/alert/template/{id} \
  -H Content-Type: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Content-Type: application/json,
  X-Auth-Token: string
}

r = requests.put(https://PEM-SERVER-IP/pem/api/v17/alert/template/{id}, headers =␣
↪headers)

print(r.json())
```

```
const inputBody = {
  "name": "string",
  "description": "string",
```

(次のページに続く)

(前のページからの続き)

```
  "sql": "string",
  "applicable_on_server": "ALL",
  "default_check_frequency": 0,
  "default_history_retention": 0,
  "probe_dependency_list": {
    "deleted": [
      {
        "internal_name": "probe_name"
      }
    ],
    "added": [
      {
        "internal_name": "probe_name"
      }
    ]
  },
  "threshold_unit": "string",
  "is_system_template": false,
  "info_sql": "string"
};
const headers = {
  Content-Type:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v17/alert/template/{id},
{
  method: PUT,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

PUT /alert/template/{id}

アラートテンプレートを更新します。

**ボディパラメーター**

```
{
  "name": "string",
  "description": "string",
```

(次のページに続く)

(前のページからの続き)


```
  "sql": "string",
  "applicable_on_server": "ALL",
  "default_check_frequency": 0,
  "default_history_retention": 0,
  "probe_dependency_list": {
    "deleted": [
      {
        "internal_name": "probe_name"
      }
    ],
    "added": [
      {
        "internal_name": "probe_name"
      }
    ]
  },
  "threshold_unit": "string",
  "is_system_template": false,
  "info_sql": "string"
}
```


### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| id | path | integer | true | アラートテンプレート ID |
| X-Auth-Token | header | string | true | 承認トークン |
| body | body | *AlertTemplatePut* | true | なし |

### 応答例

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ステータス 200 | None |

### レスポンススキーマ

> **注釈**
>
> この操作には認証が必要ありません

### delete_*alert_template*{id}

#### コードサンプル

```
#  You can also use wget

curl -X DELETE https://PEM-SERVER-IP/pem/api/v17/alert/template/{id} \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  X-Auth-Token: string
}

r = requests.delete(https://PEM-SERVER-IP/pem/api/v17/alert/template/{id}, headers =␣
↪headers)

print(r.json())
```

```
const headers = {
  "X-Auth-Token": "string",
};

fetch("https://PEM-SERVER-IP/pem/api/v17/alert/template/{id}", {
  method: "DELETE",

  headers: headers,
})
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

`DELETE /alert/template/{id}`

アラートテンプレートを削除します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| id | path | integer | true | アラートテンプレート ID |
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 削除が成功しました。 | None |

### レスポンススキーマ

**注釈**

この操作には認証が必要ありません

## Export Custom Alert Templates

### *post_alert_custom_export*

### コードサンプル

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v17/alert/custom/export/ \
  -H Content-Type: application/json \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Content-Type: application/json,
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.post(https://PEM-SERVER-IP/pem/api/v17/alert/custom/export/, headers =␣
```

(次のページに続く)

(前のページからの続き)

```
↪headers)

print(r.json())
```

```
const inputBody = {
  "alert_templates": [
    0
  ]
};
const headers = {
  Content-Type:application/json,
  Accept:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v17/alert/custom/export/,
{
  method: POST,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

POST /alert/custom/export/

カスタムアラートテンプレートをエクスポートします。

**ボディパラメーター**

```
{
  "alert_templates": [0]
}
```

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| X-Auth-Token | header | string | true | 承認トークン |

次のページに続く

表 386 – 前のページからの続き

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| body | body | *CustomAlertTemplateExportData* | true | なし |

**応答例**


```
{
  "version": 0,
  "items": [
    {
      "version": 1,
      "alert_templates": [
        {
          "name": "string",
          "description": "string",
          "reference_id": "string",
          "default_history_retention": 0,
          "object_type": 0,
          "sql": "string",
          "applicable_on_server": "ALL",
          "default_check_frequency": 0,
          "threshold_unit": "string",
          "is_system_template": true,
          "is_auto_create": true,
          "thresholds": [0],
          "operator": ">",
          "high_threshold_value": "string",
          "medium_threshold_value": "string",
          "low_threshold_value": "string",
          "info_sql": "string",
          "probes": [
            {
              "probe_name": "string",
              "collection_method": "w",
              "enabled": true,
              "target_type": 0,
              "any_server_version": true,
              "any_extension_version": true,
              "extension_name": "string",
              "interval": 0,
              "lifetime": 0,
              "discard_history": true,
```


(次のページに続く)

(前のページからの続き)


```
            "platform": "unix",
            "probe_code": "string",
            "probe_columns": [
              {
                "pc_name": "string",
                "pc_data_type": "bigint",
                "pc_unit": "string",
                "pc_col_type": "m",
                "pc_graphable": true,
                "pc_calc_pit": true,
                "pc_pit_default": true
              }
            ],
            "alternate_code": [
              {
                "server_version_id": 0,
                "server_probe_code": "string",
                "extension_version": 0
              }
            ],
            "internal_name": "string"
          }
        ]
      }
    ]
  }
]
}
```


### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | Inline |

### レスポンススキーマ

ステータスコード **200**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » version | integer | false | none | エクスポートバージョン ID |
| » items | *Platform Compatibility* ] | false | none | なし |
| » version | integer | false | none | アラートテンプレートのエクスポートのバージョン |
| » alert_templates | *Platform Compatibility* ] | false | none | なし |
| » name | string | false | none | アラートテンプレートの名前。注 このプロパティは、POST 要求の必須パラメーターです。 |
| » description | string | false | none | アラートテンプレートの説明。注 このプロパティは、POST 要求の必須パラメーターです。 |
| » reference_id | string | false | none | アラートテンプレートの内部識別子 |
| » » » default_history_retention | integer | false | none | このタイプのアラートの履歴保持期間を日単位で定義します。注 このプロパティは、POST 要求の必須パラメーターです。 |

次のページに続く

表 388 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » object_type | integer | false | none | このアラートテンプレートを適用するオブジェクトのタイプを定義します。<ul><li>システム全体のアラートテンプレートは50です。</li><li>エージェントに適用されるアラートテンプレートは100です。</li> <li>管理対象サーバーに適用されるアラートテンプレートは200です。</li><li>特定の監視対象サーバーのデータベースに適用されるアラートテンプレートは300です。</li><li>管理対象サーバーの特定のデータベースのスキーマに適用されるアラートテンプレートは400です。</li><li>特定のスキーマのテーブルに適用されるアラートテンプレート</li><li>監視対象サーバー上のデータベースの特定のスキーマのインデックスに適用されるアラートテンプレートは600です。>.</li><li>監視対象サーバー上のデータベースの特定のスキーマのシーケンスに適用できるアラートテンプレートは700です。</li><li>に適用できるアラートテンプレート監視対象サーバー上のデータベースの特定のスキーマのファンクションは800で |

表 388 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » sql | string | false | none | しきい値に到達したことを示す数値を取得するための SQL クエリー。パラメーター化される場合があります。注 このプロパティは、POST 要求の必須パラメーターです。 |
| » applicable_on_server | string | false | none | テンプレートを適用する監視対象サーバーの種類を指定します。使用可能な値は、ALL、POSTGRES_SERVER、ADVANCED_SERVER です。注 このプロパティは、POST 要求の必須パラメーターです。 |
| » » » default_check_frequency | integer | false | none | このテンプレートが定義するアラートをチェックする頻度分単位を指定します。注 このプロパティは、POST 要求の必須パラメーターです。 |
| » threshold_unit | string | false | none | MB/kB/秒/日/... などの、しきい値の計算に使用される単位 この値は、GUI でのみ使用されます。 |
| » is_system_template | boolean | false | none | 指定されたテンプレートがシステム定義であるかユーザー定義であるかを判断します。注このプロパティは、POST/PUT 要求には適用されません。 |
| » is_auto_create | boolean | false | none | エージェントまたはサーバーが PEM に追加されたらすぐにアラートを作成するかどうかを決定します。 |

次のページに続く

表 388 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » thresholds | [number] | false | none | デフォルトのしきい値。しきい値配列の値は、演算子に基づいて並べ替えられた順序である必要があります。 |
| » operator | string | false | none | しきい値を比較する演算子 |
| » » » high_threshold_value | string | false | none | 高しきい値 |
| » » » medium_threshold_value | string | false | none | 中しきい値 |
| » low_threshold_value | string | false | none | 低しきい値 |
| » info_sql | string | false | none | ダッシュボードのアラートに関する詳細情報を提供する SQL クエリーを定義します。 |
| » probes | [allOf] | false | none | なし |

*allOf*

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » _anonymous_ | *CustomProbe-PostData* | false | none | なし |
| » probe_name | string | false | none | プローブの名前 |

次のページに続く

表 389 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » collection_method | string | false | none | 統計収集の方法を定義します。 PEM は、カスタムメイドのプローブの 3 つの方法 SQL、WMI、またはバッチ/シェルスクリプトをサポートしています。<ul><li>'w' は WMI スクリプトを表します。このタイプのスクリプトは Windows プラットフォームにのみ適用され、target_type は 100エージェント用です。</li><li>'b' はバッチ/シェルスクリプトを表します。このスクリプトの'target_type' は常に 100エージェントです。このスクリプトは、 Windows ではバッチスクリプトとして、他のプラットフォームではシェルスクリプトとして動作します。</li><li>'s' は SQL スクリプトの略です。 'target_type' は常に 200サーバー、 300データベース、 400スキーマ、 500です。テーブル、 600 インデックス、 700 シーケンス、 800 ファンクション、 900 ビュー) または1000拡張子。</li></ul> |

次のページに続く

表 389 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| ➤ enabled | boolean | false | none | プローブがデフォルトで有効/無効かどうかを決定します。 |

次のページに続く

表 389 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| ➤ target_type | integer | false | none | プローブが適用されるオブジェクトのタイプを定義します。<ul><li>target_type 100のプローブは、エージェントの統計を収集します。</li><li>target_type 200のプローブは、収集を担当します。</li><li>target_type 300のプローブは、データベースの統計を収集します。</li><li>target_type 400<のプローブ/b>は、スキーマの統計を収集します。</li><li>target_type 500のプローブは、テーブルの統計を収集します。</li><li>次のプローブtarget_type 600は、インデックスの統計を収集します。</li><li>target_type 700のプローブは、シーケンスの統計を収集します。</li> <li>target_type 800のプローブは、ファンクションの統計を収集します。</li><li>target_type 900のプローブは、ファンクションの統計を収集します。view.</li><li>target_type 1000のプローブは、エクステンの統計を収集します。</li></ul> |

次のページに続く

表 389 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » » » » » any_server_version | boolean | false | none | このプローブは Postgres バリアントのバージョンに適用されるかどうかを判断します。詳細については、ドキュメントを参照してください。 |
| » » » » » any_extension_version | boolean | false | none | このプローブは拡張バリアントのバージョンに適用されるかどうかを判断します。詳細については、ドキュメントを参照してください。 |
| ➽ extension_name | string | false | none | 拡張機能の名前。<ul>有効な名前は次のとおりです。 edbspl 、 plpgsql 、 hstore 、 pldbgapi 、 edb_dblink_oci 、 edb_dblink_libpq 、 sslutils .</ul> |
| ➽ interval | integer | false | none | プローブが統計を収集する頻度を秒単位で定義します。10 以上である必要があります |
| ➽ lifetime | integer | false | none | 履歴データの保持期間を日単位で定義します。有効な範囲は 1〜365 です。 'discard_history' が true に設定されている場合、この値は効果がありません。 |
| ➽ discard_history | boolean | false | none | 履歴データを破棄するかどうかを定義します。'discard_history' が false に設定されている場合、このプローブの履歴データは保存されません。 |

次のページに続く

表 389 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| ➤ platform | string | false | none | このプローブを実行できるプラットフォームを定義します。この値は、エージェントレベルのプローブにのみ適用されます。 |
| ➤ probe_code | string | false | none | コードの定義ごとに統計を収集するためのSQL/Batch スクリプト/シェルスクリプト/WMI スクリプト。 |
| ➤ probe_columns | [object] | false | none | なし |
| ➤ pc_name | string | false | none | カラム名 |
| ➤ pc_data_type | string | false | none | 列のデータ型。サポートされているデータ型は次のとおりです。<ul><ul>bigint</ul><ul>char</ul><ul>10 進 数</ul><ul>倍 精 度</ul><ul>整 数</ul><ul>数 値</ul><ul>実数</ul><ul>テ キ ス ト</ul><ul>text[]</ul><ul>timestamp</ul><ul>t/ul> |
| ➤ pc_unit | string | false | none | 列の単位を返します。 |
| ➤ pc_col_type | string | false | none | 内部テーブルの主キーの一部として列を返すか、否か。 'k' はキー列を表し、'm' はメンバー/ノーマル列を表します。 |
| ➤ pc_graphable | boolean | false | none | 列のデータをチャートにグラフ化できるかどうかを決定します。 |
| ➤ pc_calc_pit | boolean | false | none | サーバーがこの列のメトリックデータのポイントインタイムを計算するかどうかを決定します。 |

次のページに続く

表 389 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| ➽ pc_pit_default | boolean | false | none | メトリックデータをデフォルトでポイントインタイムで保存するかどうかを決定します。詳細については、ドキュメントの'Custom Probes'セクションを参照してください。 |
| ➽ alternate_code | *Platform Compatibility* ] | false | none | なし |
| » server_version_id | integer | false | none | 監視対象サーバーのサーバーバージョン ID<table class="model"><thead><tr><th>server_version Server</th></tr></thead><thead /> <tbody><tr><td>10905</td><td>PostgreSQL 9.5</td></tr><tr><td>10906</td><td>PostgreS 9.6</td></tr ><tr><td>11000</td><td>PostgreSQL 10</td></tr><tr><td>11100</td><td>PostgreS 11</td></tr>< tr><td>11200</td><td>PostgreSQL 12</td></tr><tr><td>11300</td><td>PostgreS 13</td></tr><tr> <td>20905</td><td>Postgres Plus Advanced Server 9.5</td></tr><tr><td>20906</td><td>Postgres Plus Advanced Server 9.6</td></tr ><tr><td>21000</td><td> EDB Postgres Advanced Server 10</td></tr><tr><td>21100</td><td> EDB Postgres Advanced Server 11</td ></tr><tr><td>21200 </td><td> EDB Postgres Advanced Server 12</td></tr><tr><td>21300 </td><td> EDB Postgres Advanced Server 13</td></tr></tbody></table> |

次のページに続く

表 389 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » » » » » » server_probe_code | string | false | none | 指定されたサーバーバージョンで実行する SQL クエリー |
| » extension_version | integer | false | none | 拡張機能のバージョンを指定します。拡張ターゲットタイプに有効な |

*and*

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » _anonymous_ | object | false | none | なし |
| » internal_name | string | false | none | プローブがエクスポートされたシステムからのプローブの内部名 |

### 列挙値

| Property | Value |
|---|---|
| applicable_on_server | ALL |
| applicable_on_server | POSTGRES_SERVER |
| applicable_on_server | ADVANCED_SERVER |
| operator | > |
| operator | < |
| collection_method | w |
| collection_method | b |
| collection_method | s |
| platform | unix |
| platform | windows |
| pc_data_type | bigint |
| pc_data_type | char |
| pc_data_type | decimal |
| pc_data_type | double precision |
| pc_data_type | integer |
| pc_data_type | numeric |
| pc_data_type | real |
| pc_data_type | text |
| pc_data_type | text[] |
| pc_data_type | timestamp |
| pc_data_type | timestamptz |
| pc_col_type | m |

次のページに続く

表 391 – 前のページからの続き

| Property | Value |
|---|---|
| pc_col_type | k |

**注釈**

この操作には認証が必要ありません

### Import Custom Alert Templates

#### post_alert_custom_import

**コードサンプル**

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v17/alert/custom/import/ \
  -H Content-Type: application/json \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Content-Type: application/json,
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.post(https://PEM-SERVER-IP/pem/api/v17/alert/custom/import/, headers =␣
↪headers)

print(r.json())
```

```
const inputBody = {
  "version": 1,
  "skip_overwrite": true,
  "skip_overwrite_probe": true,
  "alert_templates": [
    {
      "name": "string",
      "description": "string",
      "reference_id": "string",
      "default_history_retention": 0,
      "object_type": 0,
```

(次のページに続く)

(前のページからの続き)


```
    "sql": "string",
    "applicable_on_server": "ALL",
    "default_check_frequency": 0,
    "threshold_unit": "string",
    "is_system_template": true,
    "is_auto_create": true,
    "thresholds": [
      0
    ],
    "operator": ">",
    "high_threshold_value": "string",
    "medium_threshold_value": "string",
    "low_threshold_value": "string",
    "info_sql": "string",
    "probes": [
      {
        "probe_name": "string",
        "collection_method": "w",
        "enabled": true,
        "target_type": 0,
        "any_server_version": true,
        "any_extension_version": true,
        "extension_name": "string",
        "interval": 0,
        "lifetime": 0,
        "discard_history": true,
        "platform": "unix",
        "probe_code": "string",
        "probe_columns": [
          {
            "pc_name": "string",
            "pc_data_type": "bigint",
            "pc_unit": "string",
            "pc_col_type": "m",
            "pc_graphable": true,
            "pc_calc_pit": true,
            "pc_pit_default": true
          }
        ],
        "alternate_code": [
          {
            "server_version_id": 0,
            "server_probe_code": "string",
            "extension_version": 0
          }
```


(次のページに続く)

(前のページからの続き)

```
          ],
          "internal_name": "string"
        }
      ]
    }
  ]
};
const headers = {
  Content-Type:application/json,
  Accept:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v17/alert/custom/import/,
{
  method: POST,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

POST /alert/custom/import/

エクスポートされたアラートテンプレートを使用して、カスタムアラートテンプレートをインポートします。

**ボディパラメーター**

```
{
  "version": 1,
  "skip_overwrite": true,
  "skip_overwrite_probe": true,
  "alert_templates": [
    {
      "name": "string",
      "description": "string",
      "reference_id": "string",
      "default_history_retention": 0,
      "object_type": 0,
      "sql": "string",
      "applicable_on_server": "ALL",
      "default_check_frequency": 0,
      "threshold_unit": "string",
```

(次のページに続く)

(前のページからの続き)

```
    "is_system_template": true,
    "is_auto_create": true,
    "thresholds": [0],
    "operator": ">",
    "high_threshold_value": "string",
    "medium_threshold_value": "string",
    "low_threshold_value": "string",
    "info_sql": "string",
    "probes": [
      {
        "probe_name": "string",
        "collection_method": "w",
        "enabled": true,
        "target_type": 0,
        "any_server_version": true,
        "any_extension_version": true,
        "extension_name": "string",
        "interval": 0,
        "lifetime": 0,
        "discard_history": true,
        "platform": "unix",
        "probe_code": "string",
        "probe_columns": [
          {
            "pc_name": "string",
            "pc_data_type": "bigint",
            "pc_unit": "string",
            "pc_col_type": "m",
            "pc_graphable": true,
            "pc_calc_pit": true,
            "pc_pit_default": true
          }
        ],
        "alternate_code": [
          {
            "server_version_id": 0,
            "server_probe_code": "string",
            "extension_version": 0
          }
        ],
        "internal_name": "string"
      }
    ]
  }
]
```

(次のページに続く)

(前のページからの続き)

```
}
```

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| X-Auth-Token | header | string | true | 承認トークン |
| body | body | *Custom-AlertTemplateImport-PostData* | true | なし |

### 応答例

```
[
  {
    "name": "string",
    "status": "string",
    "msg": "string"
  }
]
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ステータス 200 | Inline |

### レスポンススキーマ

ステータスコード **200**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » name | string | true | none | なし |
| » status | string | true | none | なし |
| » msg | string | true | none | なし |

**注釈**

この操作には認証が必要ありません

### Export Custom Charts

#### post_*chart_custom_export*

コードサンプル

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v17/chart/custom/export/ \
  -H Content-Type: application/json \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Content-Type: application/json,
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.post(https://PEM-SERVER-IP/pem/api/v17/chart/custom/export/, headers =␣
↪headers)

print(r.json())
```

```
const inputBody = {
  "charts": [
    0
  ]
};
const headers = {
  Content-Type:application/json,
  Accept:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v17/chart/custom/export/,
{
  method: POST,
  body: inputBody,
  headers: headers
})
.then(function(res) {
```

(次のページに続く)

(前のページからの続き)

```
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

POST /chart/custom/export/

カスタムチャートをエクスポートします。

### ボディパラメーター

```
{
  "charts": [0]
}
```

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| X-Auth-Token | header | string | true | 承認トークン |
| body | body | *CustomChartExportData* | true | なし |

### 応答例

```
{
  "version": 0,
  "items": [
    {
      "version": 1,
      "alert_templates": [
        {
          "name": "string",
          "description": "string",
          "reference_id": "string",
          "default_history_retention": 0,
          "object_type": 0,
          "sql": "string",
          "applicable_on_server": "ALL",
          "default_check_frequency": 0,
          "threshold_unit": "string",
          "is_system_template": true,
          "is_auto_create": true,
          "thresholds": [0],
          "operator": ">",
```

(次のページに続く)

(前のページからの続き)


```
          "high_threshold_value": "string",
          "medium_threshold_value": "string",
          "low_threshold_value": "string",
          "info_sql": "string",
          "probes": [
            {
              "probe_name": "string",
              "collection_method": "w",
              "enabled": true,
              "target_type": 0,
              "any_server_version": true,
              "any_extension_version": true,
              "extension_name": "string",
              "interval": 0,
              "lifetime": 0,
              "discard_history": true,
              "platform": "unix",
              "probe_code": "string",
              "probe_columns": [
                {
                  "pc_name": "string",
                  "pc_data_type": "bigint",
                  "pc_unit": "string",
                  "pc_col_type": "m",
                  "pc_graphable": true,
                  "pc_calc_pit": true,
                  "pc_pit_default": true
                }
              ],
              "alternate_code": [
                {
                  "server_version_id": 0,
                  "server_probe_code": "string",
                  "extension_version": 0
                }
              ],
              "internal_name": "string"
            }
          ]
        }
      ]
    }
  ]
}
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | Inline |

### レスポンススキーマ

ステータスコード **200**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » version | integer | false | none | エクスポートバージョン ID |
| » items | *Platform Compatibility* ] | false | none | なし |
| » version | integer | false | none | アラートテンプレートのエクスポートのバージョン |
| » alert_templates | *Platform Compatibility* ] | false | none | なし |
| » name | string | false | none | アラートテンプレートの名前。注 このプロパティは、POST 要求の必須パラメーターです。 |
| » description | string | false | none | アラートテンプレートの説明。注 このプロパティは、POST 要求の必須パラメーターです。 |
| » reference_id | string | false | none | アラートテンプレートの内部識別子 |
| » » » default_history_retention | integer | false | none | このタイプのアラートの履歴保持期間を日単位で定義します。注 このプロパティは、POST 要求の必須パラメーターです。 |

次のページに続く

表 397 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » object_type | integer | false | none | このアラートテンプレートを適用するオブジェクトのタイプを定義します。<ul><li>システム全体のアラートテンプレートは50です。</li><li>エージェントに適用されるアラートテンプレートは100です。</li> <li>管理対象サーバーに適用されるアラートテンプレートは200です。</li><li>特定の監視対象サーバーのデータベースに適用されるアラートテンプレートは300です。</li><li>管理対象サーバーの特定のデータベースのスキーマに適用されるアラートテンプレートは400です。</li><li>特定のスキーマのテーブルに適用されるアラートテンプレート</li><li>監視対象サーバー上のデータベースの特定のスキーマのインデックスに適用されるアラートテンプレートは600です。>.</li><li>監視対象サーバー上のデータベースの特定のスキーマのシーケンスに適用できるアラートテンプレートは700です。</li><li>に適用できるアラートテンプレート監視対象サーバー上のデータベースの特定のスキーマのファンクションは800で |

表 397 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » sql | string | false | none | しきい値に到達したことを示す数値を取得するための SQL クエリー。パラメーター化される場合があります。注 このプロパティは、POST 要求の必須パラメーターです。 |
| » applicable_on_server | string | false | none | テンプレートを適用する監視対象サーバーの種類を指定します。使用可能な値は、ALL、POSTGRES_SERVER、ADVANCED_SERVER です。注 このプロパティは、POST 要求の必須パラメーターです。 |
| » » » default_check_frequency | integer | false | none | このテンプレートが定義するアラートをチェックする頻度分単位を指定します。注 このプロパティは、POST 要求の必須パラメーターです。 |
| » threshold_unit | string | false | none | MB/kB/秒/日/... などの、しきい値の計算に使用される単位 この値は、GUI でのみ使用されます。 |
| » is_system_template | boolean | false | none | 指定されたテンプレートがシステム定義であるかユーザー定義であるかを判断します。注このプロパティは、POST/PUT 要求には適用されません。 |
| » is_auto_create | boolean | false | none | エージェントまたはサーバーが PEM に追加されたらすぐにアラートを作成するかどうかを決定します。 |

次のページに続く

表 397 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » thresholds | [number] | false | none | デフォルトのしきい値。しきい値配列の値は、演算子に基づいて並べ替えられた順序である必要があります。 |
| » operator | string | false | none | しきい値を比較する演算子 |
| » » » high_threshold_value | string | false | none | 高しきい値 |
| » » » medium_threshold_value | string | false | none | 中しきい値 |
| » low_threshold_value | string | false | none | 低しきい値 |
| » info_sql | string | false | none | ダッシュボードのアラートに関する詳細情報を提供する SQL クエリーを定義します。 |
| » probes | [allOf] | false | none | なし |

*allOf*

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » _anonymous_ | *CustomProbe-PostData* | false | none | なし |
| » probe_name | string | false | none | プローブの名前 |

次のページに続く

表 398 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » collection_method | string | false | none | 統計収集の方法を定義します。 PEM は、カスタムメイドのプローブの 3 つの方法 SQL、WMI、またはバッチ/シェルスクリプトをサポートしています。<ul><li>'w' は WMI スクリプトを表します。このタイプのスクリプトは Windows プラットフォームにのみ適用され、target_type は 100エージェント用です。</li><li>'b' はバッチ/シェルスクリプトを表します。このスクリプトの'target_type' は常に 100エージェントです。このスクリプトは、 Windows ではバッチスクリプトとして、他のプラットフォームではシェルスクリプトとして動作します。</li><li>'s' は SQL スクリプトの略です。 'target_type' は常に 200サーバー、 300データベース、 400スキーマ、 500です。テーブル、 600 インデックス、 700 シーケンス、 800 ファンクション、 900 ビュー) または1000拡張子。</li></ul> |

次のページに続く

表 398 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| ➤ enabled | boolean | false | none | プローブがデフォルトで有効/無効かどうかを決定します。 |

次のページに続く

表 398 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| ➤ target_type | integer | false | none | プローブが適用されるオブジェクトのタイプを定義します。<ul><li>target_type 100のプローブは、エージェントの統計を収集します。</li><li>target_type 200のプローブは、収集を担当します。</li><li>target_type 300のプローブは、データベースの統計を収集します。</li><li>target_type 400<のプローブ/b>は、スキーマの統計を収集します。</li><li>target_type 500のプローブは、テーブルの統計を収集します。</li><li>次のプローブtarget_type 600は、インデックスの統計を収集します。</li><li>target_type 700のプローブは、シーケンスの統計を収集します。</li> <li>target_type 800のプローブは、ファンクションの統計を収集します。</li><li>target_type 900のプローブは、ファンクションの統計を収集します。view.</li><li>target_type 1000のプローブは、エクステンの統計を収集します。</li></ul> |

次のページに続く

表 398 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » » » » » any_server_version | boolean | false | none | このプローブは Postgres バリアントのバージョンに適用されるかどうかを判断します。詳細については、ドキュメントを参照してください。 |
| » » » » » any_extension_version | boolean | false | none | このプローブは拡張バリアントのバージョンに適用されるかどうかを判断します。詳細については、ドキュメントを参照してください。 |
| ➢ extension_name | string | false | none | 拡張機能の名前。<ul>有効な名前は次のとおりです。 edbspl 、 plpgsql 、 hstore 、 pldbgapi 、 edb_dblink_oci 、 edb_dblink_libpq 、 sslutils .</ul> |
| ➢ interval | integer | false | none | プローブが統計を収集する頻度を秒単位で定義します。10 以上である必要があります |
| ➢ lifetime | integer | false | none | 履歴データの保持期間を日単位で定義します。有効な範囲は 1〜365 です。 'discard_history' が true に設定されている場合、この値は効果がありません。 |
| ➢ discard_history | boolean | false | none | 履歴データを破棄するかどうかを定義します。 'discard_history' が false に設定されている場合、このプローブの履歴データは保存されません。 |

次のページに続く

表 398 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| ➤ platform | string | false | none | このプローブを実行できるプラットフォームを定義します。この値は、エージェントレベルのプローブにのみ適用されます。 |
| ➤ probe_code | string | false | none | コードの定義ごとに統計を収集するためのSQL/Batch スクリプト/シェルスクリプト/WMI スクリプト。 |
| ➤ probe_columns | [object] | false | none | なし |
| ➤ pc_name | string | false | none | カラム名 |
| ➤ pc_data_type | string | false | none | 列のデータ型。サポートされているデータ型は次のとおりです。<ul><ul>bigint</ul><ul>char</ul><ul>10 進 数</ul><ul>倍 精 度</ul><ul>整 数</ul><ul>数 値</ul><ul>実数</ul><ul>テ キ ス ト</ul><ul>text[]</ul><ul>timestamp</ul><ul>t/ul> |
| ➤ pc_unit | string | false | none | 列の単位を返します。 |
| ➤ pc_col_type | string | false | none | 内部テーブルの主キーの一部として列を返すか、否か。 'k' はキー列を表し、'm' はメンバー/ノーマル列を表します。 |
| ➤ pc_graphable | boolean | false | none | 列のデータをチャートにグラフ化できるかどうかを決定します。 |
| ➤ pc_calc_pit | boolean | false | none | サーバーがこの列のメトリックデータのポイントインタイムを計算するかどうかを決定します。 |

次のページに続く

表 398 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| ➧ pc_pit_default | boolean | false | none | メトリックデータをデフォルトでポイントインタイムで保存するかどうかを決定します。詳細については、ドキュメントの'Custom Probes'セクションを参照してください。 |
| ➧ alternate_code | *Platform Compatibility* ] | false | none | なし |
| » server_version_id | integer | false | none | 監視対象サーバーのサーバーバージョン ID<table class="model"><thead><tr><th>server_version Server</th></tr></thead><thead /> <tbody><tr><td>10905</td><td>PostgreSQL 9.5</td></tr><tr><td>10906</td><td>PostgreS 9.6</td></tr ><tr><td>11000</td><td>PostgreSQL 10</td></tr><tr><td>11100</td><td>PostgreS 11</td></tr>< tr><td>11200</td><td>PostgreSQL 12</td></tr><tr><td>11300</td><td>PostgreS 13</td></tr><tr> <td>20905</td><td>Postgres Plus Advanced Server 9.5</td></tr><tr><td>20906</td><td>Postgres Plus Advanced Server 9.6</td></tr ><tr><td>21000</td><td> EDB Postgres Advanced Server 10</td></tr><tr><td>21100</td><td> EDB Postgres Advanced Server 11</td ></tr><tr><td>21200 </td><td> EDB Postgres Advanced Server 12</td></tr><tr><td>21300 </td><td> EDB Postgres Advanced Server 13</td></tr></tbody></table> |

次のページに続く

表 398 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » » » » » » server_probe_code | string | false | none | 指定されたサーバーバージョンで実行する SQL クエリー |
| » extension_version | integer | false | none | 拡張機能のバージョンを指定します。拡張ターゲットタイプに有効な |

*and*

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » _anonymous_ | object | false | none | なし |
| » internal_name | string | false | none | プローブがエクスポートされたシステムからのプローブの内部名 |

### 列挙値

| Property | Value |
|---|---|
| applicable_on_server | ALL |
| applicable_on_server | POSTGRES_SERVER |
| applicable_on_server | ADVANCED_SERVER |
| operator | > |
| operator | < |
| collection_method | w |
| collection_method | b |
| collection_method | s |
| platform | unix |
| platform | windows |
| pc_data_type | bigint |
| pc_data_type | char |
| pc_data_type | decimal |
| pc_data_type | double precision |
| pc_data_type | integer |
| pc_data_type | numeric |
| pc_data_type | real |
| pc_data_type | text |
| pc_data_type | text[] |
| pc_data_type | timestamp |
| pc_data_type | timestamptz |
| pc_col_type | m |

次のページに続く

表 400 – 前のページからの続き

| Property | Value |
|---|---|
| pc_col_type | k |

**注釈**

この操作には認証が必要ありません

### Import Custom Charts

#### post_*chart_custom_import*

**コードサンプル**

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v17/chart/custom/import/ \
  -H Content-Type: application/json \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Content-Type: application/json,
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.post(https://PEM-SERVER-IP/pem/api/v17/chart/custom/import/, headers =␣
↪headers)

print(r.json())
```

```
const inputBody = {
  "version": 1,
  "skip_overwrite": true,
  "skip_overwrite_probe": true,
  "charts": [
    {
      "chart_title": "string",
      "chart_type": "string",
      "chart_category": "string",
      "chart_description": "string",
      "reference_id": "string",
```

(次のページに続く)

(前のページからの続き)

```
      "chart_level": 0,
      "chart_line_ext": [
        0
      ],
      "chart_line_ext_metric": "string",
      "chart_line_ext_metric_options": [
        0
      ],
      "chart_line_ext_opt": "string",
      "chart_line_ext_val": "string",
      "chart_line_extrapolated_type": "string",
      "chart_line_points": 0,
      "chart_line_span": [
        0
      ],
      "chart_refresh": 0,
      "espan": 0,
      "line_span": 0,
      "sel_metrics_L": [
        {}
      ],
      "sel_metrics_T": [
        {}
      ],
      "sel_metrics_C": [
        {}
      ],
      "shared": [
        0
      ],
      "shared_all": true
    }
  ]
};
const headers = {
  Content-Type:application/json,
  Accept:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v17/chart/custom/import/,
{
  method: POST,
  body: inputBody,
  headers: headers
```

(次のページに続く)

(前のページからの続き)

```
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

POST /chart/custom/import/

エクスポートされたチャートを使用して、カスタムアラートテンプレートをインポートします。

**ボディパラメーター**

```
{
  "version": 1,
  "skip_overwrite": true,
  "skip_overwrite_probe": true,
  "charts": [
    {
      "chart_title": "string",
      "chart_type": "string",
      "chart_category": "string",
      "chart_description": "string",
      "reference_id": "string",
      "chart_level": 0,
      "chart_line_ext": [0],
      "chart_line_ext_metric": "string",
      "chart_line_ext_metric_options": [0],
      "chart_line_ext_opt": "string",
      "chart_line_ext_val": "string",
      "chart_line_extrapolated_type": "string",
      "chart_line_points": 0,
      "chart_line_span": [0],
      "chart_refresh": 0,
      "espan": 0,
      "line_span": 0,
      "sel_metrics_L": [{}],
      "sel_metrics_T": [{}],
      "sel_metrics_C": [{}],
      "shared": [0],
      "shared_all": true
    }
  ]
}
```

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| X-Auth-Token | header | string | true | 承認トークン |
| body | body | *CustomChartImportPostData* | true | なし |

### 応答例

```
[
  {
    "name": "string",
    "status": "string",
    "msg": "string"
  }
]
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ステータス 200 | Inline |

### レスポンススキーマ

ステータスコード **200**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » name | string | true | none | なし |
| » status | string | true | none | なし |
| » msg | string | true | none | なし |

**注釈**

この操作には認証が必要ありません

### Export Custom Dashboards

#### post_*dashboard_custom_export*

コードサンプル

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v17/dashboard/custom/export/ \
  -H Content-Type: application/json \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Content-Type: application/json,
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.post(https://PEM-SERVER-IP/pem/api/v17/dashboard/custom/export/, headers␣
↪= headers)

print(r.json())
```

```
const inputBody = {
  "dashboards": [
    0
  ]
};
const headers = {
  Content-Type:application/json,
  Accept:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v17/dashboard/custom/export/,
{
  method: POST,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

POST /dashboard/custom/export/

カスタムダッシュボードをエクスポートします。

### ボディパラメーター

```
{
  "dashboards": [0]
}
```

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| X-Auth-Token | header | string | true | 承認トークン |
| body | body | *Custom-Dashboard-ExportData* | true | なし |

### 応答例

```
{
  "version": 0,
  "items": [
    {
      "version": 1,
      "alert_templates": [
        {
          "name": "string",
          "description": "string",
          "reference_id": "string",
          "default_history_retention": 0,
          "object_type": 0,
          "sql": "string",
          "applicable_on_server": "ALL",
          "default_check_frequency": 0,
          "threshold_unit": "string",
          "is_system_template": true,
          "is_auto_create": true,
          "thresholds": [0],
          "operator": ">",
          "high_threshold_value": "string",
          "medium_threshold_value": "string",
          "low_threshold_value": "string",
          "info_sql": "string",
          "probes": [
            {
```

(次のページに続く)

(前のページからの続き)


```
            "probe_name": "string",
            "collection_method": "w",
            "enabled": true,
            "target_type": 0,
            "any_server_version": true,
            "any_extension_version": true,
            "extension_name": "string",
            "interval": 0,
            "lifetime": 0,
            "discard_history": true,
            "platform": "unix",
            "probe_code": "string",
            "probe_columns": [
              {
                "pc_name": "string",
                "pc_data_type": "bigint",
                "pc_unit": "string",
                "pc_col_type": "m",
                "pc_graphable": true,
                "pc_calc_pit": true,
                "pc_pit_default": true
              }
            ],
            "alternate_code": [
              {
                "server_version_id": 0,
                "server_probe_code": "string",
                "extension_version": 0
              }
            ],
            "internal_name": "string"
          }
        ]
      }
    ]
  }
]
}
```


応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | Inline |

### レスポンススキーマ

ステータスコード **200**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » version | integer | false | none | エクスポートバージョン ID |
| » items | *Platform Compatibility* ] | false | none | なし |
| » version | integer | false | none | アラートテンプレートのエクスポートのバージョン |
| » alert_templates | *Platform Compatibility* ] | false | none | なし |
| » name | string | false | none | アラートテンプレートの名前。注 このプロパティは、POST 要求の必須パラメーターです。 |
| » description | string | false | none | アラートテンプレートの説明。注 このプロパティは、POST 要求の必須パラメーターです。 |
| » reference_id | string | false | none | アラートテンプレートの内部識別子 |
| » » » default_history_retention | integer | false | none | このタイプのアラートの履歴保持期間を日単位で定義します。注 このプロパティは、POST 要求の必須パラメーターです。 |

次のページに続く

表 406 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » object_type | integer | false | none | このアラートテンプレートを適用するオブジェクトのタイプを定義します。<ul><li>システム全体のアラートテンプレートは50です。</li><li>エージェントに適用されるアラートテンプレートは100です。</li> <li>管理対象サーバーに適用されるアラートテンプレートは200です。</li><li>特定の監視対象サーバーのデータベースに適用されるアラートテンプレートは300です。</li><li>管理対象サーバーの特定のデータベースのスキーマに適用されるアラートテンプレートは400です。</li><li>特定のスキーマのテーブルに適用されるアラートテンプレート</li><li>監視対象サーバー上のデータベースの特定のスキーマのインデックスに適用されるアラートテンプレートは600です。>.</li><li>監視対象サーバー上のデータベースの特定のスキーマのシーケンスに適用できるアラートテンプレートは700です。</li><li>に適用できるアラートテンプレート監視対象サーバー上のデータベースの特定のスキーマのファンクションは800で |

表 406 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » sql | string | false | none | しきい値に到達したことを示す数値を取得するための SQL クエリー。パラメーター化される場合があります。注 このプロパティは、POST 要求の必須パラメーターです。 |
| » applicable_on_server | string | false | none | テンプレートを適用する監視対象サーバーの種類を指定します。使用可能な値は、ALL、POSTGRES_SERVER、ADVANCED_SERVER です。注 このプロパティは、POST 要求の必須パラメーターです。 |
| » » » default_check_frequency | integer | false | none | このテンプレートが定義するアラートをチェックする頻度分単位を指定します。注 このプロパティは、POST 要求の必須パラメーターです。 |
| » threshold_unit | string | false | none | MB/kB/秒/日/... などの、しきい値の計算に使用される単位 この値は、GUI でのみ使用されます。 |
| » is_system_template | boolean | false | none | 指定されたテンプレートがシステム定義であるかユーザー定義であるかを判断します。注このプロパティは、POST/PUT 要求には適用されません。 |
| » is_auto_create | boolean | false | none | エージェントまたはサーバーが PEM に追加されたらすぐにアラートを作成するかどうかを決定します。 |

次のページに続く

表 406 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » thresholds | [number] | false | none | デフォルトのしきい値。しきい値配列の値は、演算子に基づいて並べ替えられた順序である必要があります。 |
| » operator | string | false | none | しきい値を比較する演算子 |
| » » » high_threshold_value | string | false | none | 高しきい値 |
| » » » medium_threshold_value | string | false | none | 中しきい値 |
| » low_threshold_value | string | false | none | 低しきい値 |
| » info_sql | string | false | none | ダッシュボードのアラートに関する詳細情報を提供する SQL クエリーを定義します。 |
| » probes | [allOf] | false | none | なし |

*allOf*

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » _anonymous_ | *CustomProbe-PostData* | false | none | なし |
| » probe_name | string | false | none | プローブの名前 |

次のページに続く

表 407 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » collection_method | string | false | none | 統計収集の方法を定義します。 PEM は、カスタムメイドのプローブの 3 つの方法 SQL、WMI、またはバッチ/シェルスクリプトをサポートしています。<ul><li>'w' は WMI スクリプトを表します。このタイプのスクリプトは Windows プラットフォームにのみ適用され、target_type は 100エージェント用です。</li><li>'b' はバッチ/シェルスクリプトを表します。このスクリプトの'target_type' は常に 100エージェントです。このスクリプトは、 Windows ではバッチスクリプトとして、他のプラットフォームではシェルスクリプトとして動作します。</li><li>'s' は SQL スクリプトの略です。 'target_type' は常に 200サーバー、 300データベース、 400スキーマ、 500です。テーブル、 600 インデックス、 700 シーケンス、 800 ファンクション、 900 ビュー) または1000拡張子。</li></ul> |

次のページに続く

表 407 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| ➤ enabled | boolean | false | none | プローブがデフォルトで有効/無効かどうかを決定します。 |

次のページに続く

表 407 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| ➤ target_type | integer | false | none | プローブが適用されるオブジェクトのタイプを定義します。<ul><li>target_type 100のプローブは、エージェントの統計を収集します。</li><li>target_type 200のプローブは、収集を担当します。</li><li>target_type 300のプローブは、データベースの統計を収集します。</li><li>target_type 400<のプローブ/b>は、スキーマの統計を収集します。</li><li>target_type 500のプローブは、テーブルの統計を収集します。</li><li>次のプローブtarget_type 600は、インデックスの統計を収集します。</li><li>target_type 700のプローブは、シーケンスの統計を収集します。</li> <li>target_type 800のプローブは、ファンクションの統計を収集します。</li><li>target_type 900のプローブは、ファンクションの統計を収集します。view.</li><li>target_type 1000のプローブは、エクステンの統計を収集します。</li></ul> |

次のページに続く

表 407 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » » » » » any_server_version | boolean | false | none | このプローブは Postgres バリアントのバージョンに適用されるかどうかを判断します。詳細については、ドキュメントを参照してください。 |
| » » » » » any_extension_version | boolean | false | none | このプローブは拡張バリアントのバージョンに適用されるかどうかを判断します。詳細については、ドキュメントを参照してください。 |
| ➢ extension_name | string | false | none | 拡張機能の名前。<ul>有効な名前は次のとおりです。 edbspl 、 plpgsql 、 hstore 、 pldbgapi 、 edb_dblink_oci 、 edb_dblink_libpq 、 sslutils .</ul> |
| ➢ interval | integer | false | none | プローブが統計を収集する頻度を秒単位で定義します。10 以上である必要があります |
| ➢ lifetime | integer | false | none | 履歴データの保持期間を日単位で定義します。有効な範囲は 1〜365 です。 'discard_history' が true に設定されている場合、この値は効果がありません。 |
| ➢ discard_history | boolean | false | none | 履歴データを破棄するかどうかを定義します。 'discard_history' が false に設定されている場合、このプローブの履歴データは保存されません。 |

次のページに続く

表 407 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| ➤ platform | string | false | none | このプローブを実行できるプラットフォームを定義します。この値は、エージェントレベルのプローブにのみ適用されます。 |
| ➤ probe_code | string | false | none | コードの定義ごとに統計を収集するためのSQL/Batch スクリプト/シェルスクリプト/WMI スクリプト。 |
| ➤ probe_columns | [object] | false | none | なし |
| ➤ pc_name | string | false | none | カラム名 |
| ➤ pc_data_type | string | false | none | 列のデータ型。サポートされているデータ型は次のとおりです。<ul><ul>bigint</ul><ul>char</ul><ul>10 進 数</ul><ul>倍 精 度</ul><ul>整 数</ul><ul>数 値</ul><ul>実数</ul><ul>テ キ ス ト</ul><ul>text[]</ul><ul>timestamp</ul><ul>t/ul> |
| ➤ pc_unit | string | false | none | 列の単位を返します。 |
| ➤ pc_col_type | string | false | none | 内部テーブルの主キーの一部として列を返すか、否か。 'k' はキー列を表し、'm' はメンバー/ノーマル列を表します。 |
| ➤ pc_graphable | boolean | false | none | 列のデータをチャートにグラフ化できるかどうかを決定します。 |
| ➤ pc_calc_pit | boolean | false | none | サーバーがこの列のメトリックデータのポイントインタイムを計算するかどうかを決定します。 |

次のページに続く

表 407 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| ➽ pc_pit_default | boolean | false | none | メトリックデータをデフォルトでポイントインタイムで保存するかどうかを決定します。詳細については、ドキュメントの'Custom Probes'セクションを参照してください。 |
| ➽ alternate_code | *Platform Compatibility* ] | false | none | なし |
| » server_version_id | integer | false | none | 監視対象サーバーのサーバーバージョン ID<table class="model"><thead><tr><th>server_version Server</th></tr></thead><thead /> <tbody><tr><td>10905</td><td>PostgreSQL 9.5</td></tr><tr><td>10906</td><td>PostgreS 9.6</td></tr ><tr><td>11000</td><td>PostgreSQL 10</td></tr><tr><td>11100</td><td>PostgreS 11</td></tr>< tr><td>11200</td><td>PostgreSQL 12</td></tr><tr><td>11300</td><td>PostgreS 13</td></tr><tr> <td>20905</td><td>Postgres Plus Advanced Server 9.5</td></tr><tr><td>20906</td><td>Postgres Plus Advanced Server 9.6</td></tr ><tr><td>21000</td><td> EDB Postgres Advanced Server 10</td></tr><tr><td>21100</td><td> EDB Postgres Advanced Server 11</td ></tr><tr><td>21200 </td><td> EDB Postgres Advanced Server 12</td></tr><tr><td>21300 </td><td> EDB Postgres Advanced Server 13</td></tr></tbody></table> |

次のページに続く

表 407 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » » » » » » server_probe_code | string | false | none | 指定されたサーバーバージョンで実行する SQL クエリー |
| » extension_version | integer | false | none | 拡張機能のバージョンを指定します。拡張ターゲットタイプに有効な |

*and*

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » _anonymous_ | object | false | none | なし |
| » internal_name | string | false | none | プローブがエクスポートされたシステムからのプローブの内部名 |

### 列挙値

| Property | Value |
|---|---|
| applicable_on_server | ALL |
| applicable_on_server | POSTGRES_SERVER |
| applicable_on_server | ADVANCED_SERVER |
| operator | > |
| operator | < |
| collection_method | w |
| collection_method | b |
| collection_method | s |
| platform | unix |
| platform | windows |
| pc_data_type | bigint |
| pc_data_type | char |
| pc_data_type | decimal |
| pc_data_type | double precision |
| pc_data_type | integer |
| pc_data_type | numeric |
| pc_data_type | real |
| pc_data_type | text |
| pc_data_type | text[] |
| pc_data_type | timestamp |
| pc_data_type | timestamptz |
| pc_col_type | m |

次のページに続く

表 409 – 前のページからの続き

| Property | Value |
|---|---|
| pc_col_type | k |

**注釈**

この操作には認証が必要ありません

### Import Custom Dashboards

#### post_*dashboard_custom_import*

**コードサンプル**

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v17/dashboard/custom/import/ \
  -H Content-Type: application/json \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Content-Type: application/json,
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.post(https://PEM-SERVER-IP/pem/api/v17/dashboard/custom/import/, headers
↪= headers)

print(r.json())
```

```
const inputBody = {
  "version": 1,
  "skip_overwrite": true,
  "skip_overwrite_probe": true,
  "skip_overwrite_chart": true,
  "dashboards": [
    {
      "name": "string",
      "descp": "string",
      "reference_id": "string",
      "font": "string",
```

(次のページに続く)

(前のページからの続き)

```
      "font_size": 0,
      "level": 0,
      "is_ops": true,
      "show_title": true,
      "design_layout": [
        {}
      ],
      "c_charts": [
        {}
      ],
      "shared": [
        0
      ],
      "shared_all": true
    }
  ]
};
const headers = {
  Content-Type:application/json,
  Accept:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v17/dashboard/custom/import/,
{
  method: POST,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

POST /dashboard/custom/import/

エクスポートされたダッシュボードを使用して、カスタムアラートテンプレートをインポートします。

**ボディパラメーター**

```
{
  "version": 1,
  "skip_overwrite": true,
  "skip_overwrite_probe": true,
  "skip_overwrite_chart": true,
```

(次のページに続く)

(前のページからの続き)


```
  "dashboards": [
    {
      "name": "string",
      "descp": "string",
      "reference_id": "string",
      "font": "string",
      "font_size": 0,
      "level": 0,
      "is_ops": true,
      "show_title": true,
      "design_layout": [{}],
      "c_charts": [{}],
      "shared": [0],
      "shared_all": true
    }
  ]
}
```


### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| X-Auth-Token | header | string | true | 承認トークン |
| body | body | *Custom-Dashboard-ImportPost-Data* | true | なし |

### 応答例


```
[
  {
    "name": "string",
    "status": "string",
    "msg": "string"
  }
]
```


### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ステータス 200 | Inline |

### レスポンススキーマ

ステータスコード **200**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » name | string | true | none | なし |
| » status | string | true | none | なし |
| » msg | string | true | none | なし |

**注釈**

この操作には認証が必要ありません

## Custom Probe

### get_*probe*

#### コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/probe/ \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/probe/, headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
```

(次のページに続く)

(前のページからの続き)

```
  "X-Auth-Token": "string",
};

fetch("https://PEM-SERVER-IP/pem/api/v17/probe/", {
  method: "GET",

  headers: headers,
})
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /probe/

すべてのカスタム/システムプローブを取得します。 internal_name 、internal_name_contains 、display_name 、または display_name_contains クエリパラメーターを介したオプションのフィルタリングをサポートします。一度に指定できるフィルタは 1 つだけです。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| X-Auth-Token | header | string | true | 承認トークン |
| internal_name | query | string | false | external_name の完全な一致の大文字と小文字を区別してプローブをフィルタリングします。他のフィルタと相互排他的。 |
| internal_name_contains | query | string | false | external_name の部分文字列一致大文字と小文字を区別してプローブをフィルタリングします。他のフィルタと相互排他的。 |
| display_name | query | string | false | display_name の完全な一致の大文字と小文字を区別してプローブをフィルタリングします。他のフィルタと相互排他的。 |
| display_name_contains | query | string | false | display_name の部分文字列の一致大文字と小文字を区別してプローブをフィルタリングします。他のフィルタと相互排他的。 |

### 応答例

```
[
  {
    "probe_id": 0,
    "probe_name": "string",
    "enabled": true,
    "target_type": 0,
    "internal_name": "string",
    "is_system_probe": true,
    "any_server_version": true,
    "collection_method": "w",
    "interval": 0,
    "lifetime": 0,
    "discard_history": true,
    "platform": "unix",
    "probe_code": "string",
    "probe_columns": [
      {
        "pc_id": 0,
        "pc_name": "string",
        "pc_data_type": "bigint",
        "pc_internal_name": "string",
        "pc_unit": "string",
        "pc_col_type": "m",
        "pc_position": 0,
        "pc_graphable": true,
        "pc_calc_pit": true,
        "pc_pit_default": true
      }
    ],
    "alternate_code": [
      {
        "server_version_id": 0,
        "server_probe_code": "string",
        "extension_version": 0
      }
    ]
  }
]
```


応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | Inline |
| 400 | Bad Request | 相互排他性違反 `name`または類似の完全一致フィルタと`name_contains`または類似の部分文字列フィルタが両方とも同じ要求で指定されました。フィルタリングは、ペアのうちの 1 つのみを受け入れます。 | Inline |

### レスポンススキーマ

ステータスコード **200**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| _anonymous_ | *Platform Compatibility* ] | false | none | なし |
| » probe_id | integer | false | none | プローブの識別子 |
| » probe_name | string | false | none | プローブの名前 |
| » enabled | boolean | false | none | プローブがデフォルトで有効か無効かを決定します。 |

次のページに続く

表 415 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » target_type | integer | false | none | プローブが適用されるオブジェクトのタイプを定義します。<ul><li>target_type 100のプローブは、エージェントの統計を収集します。</li><li>target_type 200のプローブは、収集を担当します。</li><li>target_type 300のプローブは、データベースの統計を収集します。</li><li>target_type 400<のプローブ/b>は、スキーマの統計を収集します。</li><li>target_type 500のプローブは、テーブルの統計を収集します。</li><li>次のプローブtarget_type 600は、インデックスの統計を収集します。</li><li>target_type 700のプローブは、シーケンスの統計を収集します。</li> <li>target_type 800のプローブは、ファンクションの統計を収集します。</li><li>target_type 900のプローブは、ファンクションの統計を収集します。view.</li><li>target_type 1000のプローブは、エクステンの統計を収集します。</li></ul> |

次のページに続く

表 415 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » internal_name | string | false | none | プローブの内部名を定義します。 PEM システムは、プローブ統計を'pem-data' および'pemhistory' スキーマに保存するための内部テーブルを作成します。これらのテーブルの名前は、内部名と同じです。 |
| » is_system_probe | boolean | false | none | このプローブがユーザー定義であるかどうかを判断します。ユーザー定義プローブのこのフラグは常に false です。 |
| » any_server_version | boolean | false | none | このプローブは Postgres バリアントのバージョンに適用されるかどうかを判断します。詳細については、ドキュメントを参照してください。 |

次のページに続く

表 415 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » collection_method | string | false | none | 統計収集の方法を定義します。 PEM は、カスタムメイドのプローブの 3 つの方法 SQL、WMI、またはバッチ/シェルスクリプトをサポートしています。<ul><li>'w' は WMI スクリプトを表します。このタイプのスクリプトは Windows プラットフォームにのみ適用され、target_type は 100エージェント用です。</li><li>'b' はバッチ/シェルスクリプトを表します。このスクリプトの'target_type' は常に 100エージェントです。このスクリプトは、 Windows ではバッチスクリプトとして、他のプラットフォームではシェルスクリプトとして動作します。</li><li>'s' は SQL スクリプトの略です。 'target_type' は常に 200サーバー、 300データベース、 400スキーマ、 500です。テーブル、 600インデックス、 700シーケンス、 800ファンクション、または 900 view)、または1000拡張機能。 </li></ul> |

次のページに続く

表 415 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » interval | integer | false | none | プローブが統計を収集する頻度を秒単位で定義します。10 以上である必要があります |
| » lifetime | integer | false | none | 履歴データの保持期間を日単位で定義します。有効な範囲は 1〜365 です。 'discard_history' が true に設定されている場合、この値は効果がありません。 |
| » discard_history | boolean | false | none | 履歴データを破棄するかどうかを決定します。'discard_history' が false に設定されている場合、このプローブのデータの履歴は保存されません。 |
| » platform | string | false | none | このプローブを実行できるプラットフォームを定義します。この値は、エージェントレベルのプローブにのみ適用されます。 |
| » probe_code | string | false | none | コードの定義に従って統計を収集するためのSQL/Batch スクリプト/シェルスクリプト/WMI スクリプト。 |
| » probe_columns | [object] | false | none | なし |
| » pc_id | integer | false | none | 列識別子 |
| » pc_name | string | false | none | カラム名 |
| » pc_data_type | string | false | none | 列のデータ型。サポートされているデータ型は次のとおりです。<ul><ul>bigint</ul><ul>char</ul><ul>10 進 数</ul><ul>倍 精 度</ul><ul>整 数</ul><ul>数 値</ul><ul>実数</ul><ul>テ キ ス ト</ul><ul>text[]</ul><ul>timestamp</ul><ul>t/ul> |
| » pc_internal_name | string | false | none | 列の内部名を返します。 |
| » pc_unit | string | false | none | 列の単位を返します。 |

次のページに続く

表 415 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » pc_col_type | string | false | none | 列を内部テーブルの主キーの一部として返すか、否か。 'k' はキー列を表し、'm' はメンバー/ノーマル列を表します。 |
| » pc_position | integer | false | none | 内部テーブルの列の位置を返します。 |
| » pc_graphable | boolean | false | none | 列のデータをチャートにグラフ化できるかどうかを決定します。 |
| » pc_calc_pit | boolean | false | none | サーバーがこの列のメトリックデータのポイントインタイムを計算するかどうかを決定します。 |
| » pc_pit_default | boolean | false | none | メトリックをデフォルトでポイントインタイムで保存するかどうかを決定します。詳細については、ドキュメントの'Custom Probes' セクションを参照してください。 |
| » alternate_code | *Platform Compatibility* ] | false | none | なし |

次のページに続く

表 415 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » server_version_id | integer | false | none | 監視対象サーバーのサーバーバージョン ID<table class="model"><thead><tr><th>server_version Server</th></tr></thead><thead /><tbody><tr><td>10905</td><td>PostgreSQL 9.5</td></tr><tr><td>10906</td><td>PostgreS 9.6</td></tr><tr><td>11000</td><td>PostgreSQL 10</td></tr><tr><td>11100</td><td>PostgreS 11</td></tr><tr><td>11200</td><td>PostgreSQL 12</td></tr><tr><td>11300</td><td>PostgreS 13</td></tr><tr><td>20905</td><td>Postgres Plus Advanced Server 9.5</td></tr><tr><td>20906</td><td>Postgres Plus Advanced Server 9.6</td></tr><tr><td>21000</td><td>EDB Postgres Advanced Server 10</td></tr><tr><td>21100</td><td>EDB Postgres Advanced Server 11</td></tr><tr><td>21200</td><td>EDB Postgres Advanced Server 12</td></tr><tr><td>21300</td><td>EDB Postgres Advanced Server 13</td></tr></tbody></table> |
| » server_probe_code | string | false | none | 指定されたサーバーバージョンで実行する SQL クエリー |
| » extension_version | integer | false | none | 拡張機能のバージョンを指定します。拡張ターゲットタイプに有効な |

### 列挙値

| Property | Value |
|---|---|
| collection_method | w |
| collection_method | b |
| collection_method | s |
| platform | unix |
| platform | windows |
| pc_data_type | bigint |
| pc_data_type | char |
| pc_data_type | decimal |
| pc_data_type | double precision |
| pc_data_type | integer |
| pc_data_type | numeric |
| pc_data_type | real |
| pc_data_type | text |
| pc_data_type | text[] |
| pc_data_type | timestamp |
| pc_data_type | timestamptz |
| pc_col_type | m |
| pc_col_type | k |

ステータスコード **400**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » success | integer | false | none | なし |
| » errormsg | string | false | none | なし |

**注釈**

この操作には認証が必要ありません

## post_probe

### コードサンプル

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v17/probe/ \
  -H Content-Type: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Content-Type: application/json,
  X-Auth-Token: string
}

r = requests.post(https://PEM-SERVER-IP/pem/api/v17/probe/, headers = headers)

print(r.json())
```

```
const inputBody = {
  "probe_name": "string",
  "collection_method": "w",
  "enabled": true,
  "target_type": 0,
  "any_server_version": true,
  "any_extension_version": true,
  "extension_name": "string",
  "interval": 0,
  "lifetime": 0,
  "discard_history": true,
  "platform": "unix",
  "probe_code": "string",
  "probe_columns": [
    {
      "pc_name": "string",
      "pc_data_type": "bigint",
      "pc_unit": "string",
      "pc_col_type": "m",
      "pc_graphable": true,
      "pc_calc_pit": true,
      "pc_pit_default": true
    }
  ],
  "alternate_code": [
    {
      "server_version_id": 0,
      "server_probe_code": "string",
      "extension_version": 0
    }
  ]
};
const headers = {
  Content-Type:application/json,
  X-Auth-Token:string
```

(次のページに続く)

(前のページからの続き)

```
};

fetch(https://PEM-SERVER-IP/pem/api/v17/probe/,
{
  method: POST,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

POST /probe/

新しいカスタムプローブを作成します。

**ボディパラメーター**

```
{
  "probe_name": "string",
  "collection_method": "w",
  "enabled": true,
  "target_type": 0,
  "any_server_version": true,
  "any_extension_version": true,
  "extension_name": "string",
  "interval": 0,
  "lifetime": 0,
  "discard_history": true,
  "platform": "unix",
  "probe_code": "string",
  "probe_columns": [
    {
      "pc_name": "string",
      "pc_data_type": "bigint",
      "pc_unit": "string",
      "pc_col_type": "m",
      "pc_graphable": true,
      "pc_calc_pit": true,
      "pc_pit_default": true
    }
  ],
  "alternate_code": [
    {
```

(次のページに続く)

(前のページからの続き)


```
      "server_version_id": 0,
      "server_probe_code": "string",
      "extension_version": 0
    }
  ]
}
```


### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| X-Auth-Token | header | string | true | 承認トークン |
| body | body | *CustomProbePostData* | true | なし |

### 応答例

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ステータス 200 | None |

### レスポンススキーマ

**注釈**

この操作には認証が必要ありません

## get_*probe*{probe_id}

### コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/probe/{probe_id} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/probe/{probe_id}, headers =␣
↪headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch("https://PEM-SERVER-IP/pem/api/v17/probe/{probe_id}", {
  method: "GET",

  headers: headers,
})
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /probe/{probe_id}

指定されたプローブ ID のプローブ情報を取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| probe_id | path | integer | true | プローブ情報が返されるプローブ ID |
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例

```
{
  "probe_id": 0,
  "probe_name": "string",
  "enabled": true,
```

(次のページに続く)

(前のページからの続き)


```
  "target_type": 0,
  "internal_name": "string",
  "is_system_probe": true,
  "any_server_version": true,
  "collection_method": "w",
  "interval": 0,
  "lifetime": 0,
  "discard_history": true,
  "platform": "unix",
  "probe_code": "string",
  "probe_columns": [
    {
      "pc_id": 0,
      "pc_name": "string",
      "pc_data_type": "bigint",
      "pc_internal_name": "string",
      "pc_unit": "string",
      "pc_col_type": "m",
      "pc_position": 0,
      "pc_graphable": true,
      "pc_calc_pit": true,
      "pc_pit_default": true
    }
  ],
  "alternate_code": [
    {
      "server_version_id": 0,
      "server_probe_code": "string",
      "extension_version": 0
    }
  ]
}
```


応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | *CustomProbes* |

> **注釈**
>
> この操作には認証が必要ありません

### put_*probe*{probe_id}

#### コードサンプル

```
#  You can also use wget

curl -X PUT https://PEM-SERVER-IP/pem/api/v17/probe/{probe_id} \
  -H Content-Type: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Content-Type: application/json,
  X-Auth-Token: string
}

r = requests.put(https://PEM-SERVER-IP/pem/api/v17/probe/{probe_id}, headers =␣
↪headers)

print(r.json())
```

```
const inputBody = {
  "interval_min": 0,
  "interval_sec": 0,
  "lifetime": 0,
  "enabled": true,
  "probe_code": "string",
  "probe_columns": {
    "changed": [
      {
        "pc_id": 0,
        "pc_unit": "string",
        "pc_graphable": true
      }
    ]
  },
  "alternate_code": {
    "changed": [
      {
        "server_version_id": 0,
        "server_probe_code": "string",
```

(次のページに続く)

(前のページからの続き)

```
        "extension_version": 0
      }
    ],
    "deleted": [
      {
        "server_version_id": 0
      }
    ],
    "added": [
      {
        "server_version_id": 0,
        "server_probe_code": "string",
        "extension_version": 0
      }
    ]
  }
};
const headers = {
  Content-Type:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v17/probe/{probe_id},
{
  method: PUT,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

PUT /probe/{probe_id}

指定されたプローブ ID のカスタム/システムプローブ情報を更新します。

**ボディパラメーター**

```
{
  "interval_min": 0,
  "interval_sec": 0,
  "lifetime": 0,
  "enabled": true,
  "probe_code": "string",
```

(次のページに続く)

(前のページからの続き)

```
  "probe_columns": {
    "changed": [
      {
        "pc_id": 0,
        "pc_unit": "string",
        "pc_graphable": true
      }
    ]
  },
  "alternate_code": {
    "changed": [
      {
        "server_version_id": 0,
        "server_probe_code": "string",
        "extension_version": 0
      }
    ],
    "deleted": [
      {
        "server_version_id": 0
      }
    ],
    "added": [
      {
        "server_version_id": 0,
        "server_probe_code": "string",
        "extension_version": 0
      }
    ]
  }
}
```

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| probe_id | path | integer | true | プローブ情報が返されるプローブ ID |
| X-Auth-Token | header | string | true | 承認トークン |
| body | body | *CustomProbePutData* | true | なし |

### 応答例

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ステータス 200 | None |

### レスポンススキーマ

> **注釈**
>
> この操作には認証が必要ありません

### delete_*probe*{probe_id}

#### コードサンプル

```
#  You can also use wget

curl -X DELETE https://PEM-SERVER-IP/pem/api/v17/probe/{probe_id} \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  X-Auth-Token: string
}

r = requests.delete(https://PEM-SERVER-IP/pem/api/v17/probe/{probe_id}, headers =␣
↪headers)

print(r.json())
```

```
const headers = {
  "X-Auth-Token": "string",
};

fetch("https://PEM-SERVER-IP/pem/api/v17/probe/{probe_id}", {
  method: "DELETE",

  headers: headers,
})
  .then(function (res) {
    return res.json();
```

(次のページに続く)

(前のページからの続き)

```
  })
  .then(function (body) {
    console.log(body);
  });
```

```
DELETE /probe/{probe_id}
```

カスタムプローブを削除します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| probe_id | path | integer | true | プローブ情報が返されるプローブ ID |
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 削除が成功しました。 | None |

### レスポンススキーマ

**注釈**

この操作には認証が必要ありません

## Export Custom Probes

### post_probe_custom_export

### コードサンプル

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v17/probe/custom/export/ \
  -H Content-Type: application/json \
```

(次のページに続く)

(前のページからの続き)

```
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Content-Type: application/json,
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.post(https://PEM-SERVER-IP/pem/api/v17/probe/custom/export/, headers =␣
↪headers)

print(r.json())
```

```
const inputBody = {
  "probes": [
    0
  ]
};
const headers = {
  Content-Type:application/json,
  Accept:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v17/probe/custom/export/,
{
  method: POST,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

POST /probe/custom/export/

カスタムプローブをエクスポートします。

**ボディパラメーター**

```
{
  "probes": [0]
```

(次のページに続く)

(前のページからの続き)

```
}
```

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| X-Auth-Token | header | string | true | 承認トークン |
| body | body | *CustomProbeExport-Data* | true | なし |

### 応答例

```
{
  "version": 0,
  "items": [
    {
      "version": 1,
      "alert_templates": [
        {
          "name": "string",
          "description": "string",
          "reference_id": "string",
          "default_history_retention": 0,
          "object_type": 0,
          "sql": "string",
          "applicable_on_server": "ALL",
          "default_check_frequency": 0,
          "threshold_unit": "string",
          "is_system_template": true,
          "is_auto_create": true,
          "thresholds": [0],
          "operator": ">",
          "high_threshold_value": "string",
          "medium_threshold_value": "string",
          "low_threshold_value": "string",
          "info_sql": "string",
          "probes": [
            {
              "probe_name": "string",
              "collection_method": "w",
              "enabled": true,
              "target_type": 0,
              "any_server_version": true,
```

(次のページに続く)

(前のページからの続き)


```
            "any_extension_version": true,
            "extension_name": "string",
            "interval": 0,
            "lifetime": 0,
            "discard_history": true,
            "platform": "unix",
            "probe_code": "string",
            "probe_columns": [
              {
                "pc_name": "string",
                "pc_data_type": "bigint",
                "pc_unit": "string",
                "pc_col_type": "m",
                "pc_graphable": true,
                "pc_calc_pit": true,
                "pc_pit_default": true
              }
            ],
            "alternate_code": [
              {
                "server_version_id": 0,
                "server_probe_code": "string",
                "extension_version": 0
              }
            ],
            "internal_name": "string"
          }
        ]
      }
    ]
  }
]
}
```


### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | Inline |

### レスポンススキーマ

ステータスコード **200**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » version | integer | false | none | エクスポートバージョン ID |
| » items | *Platform Compatibility* ] | false | none | なし |
| » version | integer | false | none | アラートテンプレートのエクスポートのバージョン |
| » alert_templates | *Platform Compatibility* ] | false | none | なし |
| » name | string | false | none | アラートテンプレートの名前。注 このプロパティは、POST 要求の必須パラメーターです。 |
| » description | string | false | none | アラートテンプレートの説明。注 このプロパティは、POST 要求の必須パラメーターです。 |
| » reference_id | string | false | none | アラートテンプレートの内部識別子 |
| » » » default_history_retention | integer | false | none | このタイプのアラートの履歴保持期間を日単位で定義します。注 このプロパティは、POST 要求の必須パラメーターです。 |

次のページに続く

表 428 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » object_type | integer | false | none | このアラートテンプレートを適用するオブジェクトのタイプを定義します。<ul><li>システム全体のアラートテンプレートは50です。</li><li>エージェントに適用されるアラートテンプレートは100です。</li> <li>管理対象サーバーに適用されるアラートテンプレートは200です。</li><li>特定の監視対象サーバーのデータベースに適用されるアラートテンプレートは300です。</li><li>管理対象サーバーの特定のデータベースのスキーマに適用されるアラートテンプレートは400です。</li><li>特定のスキーマのテーブルに適用されるアラートテンプレート</li><li>監視対象サーバー上のデータベースの特定のスキーマのインデックスに適用されるアラートテンプレートは600です。>.</li><li>監視対象サーバー上のデータベースの特定のスキーマのシーケンスに適用できるアラートテンプレートは700です。</li><li>に適用できるアラートテンプレート監視対象サーバー上のデータベースの特定のスキーマのファンクションは800で |

表 428 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » sql | string | false | none | しきい値に到達したことを示す数値を取得するための SQL クエリー。パラメーター化される場合があります。注 このプロパティは、POST 要求の必須パラメーターです。 |
| » applicable_on_server | string | false | none | テンプレートを適用する監視対象サーバーの種類を指定します。使用可能な値は、ALL、POSTGRES_SERVER、ADVANCED_SERVER です。注 このプロパティは、POST 要求の必須パラメーターです。 |
| » » » default_check_frequency | integer | false | none | このテンプレートが定義するアラートをチェックする頻度分単位を指定します。注 このプロパティは、POST 要求の必須パラメーターです。 |
| » threshold_unit | string | false | none | MB/kB/秒/日/... などの、しきい値の計算に使用される単位 この値は、GUI でのみ使用されます。 |
| » is_system_template | boolean | false | none | 指定されたテンプレートがシステム定義であるかユーザー定義であるかを判断します。注このプロパティは、POST/PUT 要求には適用されません。 |
| » is_auto_create | boolean | false | none | エージェントまたはサーバーが PEM に追加されたらすぐにアラートを作成するかどうかを決定します。 |

次のページに続く

表 428 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » thresholds | [number] | false | none | デフォルトのしきい値。しきい値配列の値は、演算子に基づいて並べ替えられた順序である必要があります。 |
| » operator | string | false | none | しきい値を比較する演算子 |
| » » » high_threshold_value | string | false | none | 高しきい値 |
| » » » medium_threshold_value | string | false | none | 中しきい値 |
| » low_threshold_value | string | false | none | 低しきい値 |
| » info_sql | string | false | none | ダッシュボードのアラートに関する詳細情報を提供する SQL クエリーを定義します。 |
| » probes | [allOf] | false | none | なし |

*allOf*

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » _anonymous_ | *CustomProbe-PostData* | false | none | なし |
| » probe_name | string | false | none | プローブの名前 |

次のページに続く

表 429 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » collection_method | string | false | none | 統計収集の方法を定義します。 PEM は、カスタムメイドのプローブの 3 つの方法 SQL、WMI、またはバッチ/シェルスクリプトをサポートしています。<ul><li>'w' は WMI スクリプトを表します。このタイプのスクリプトは Windows プラットフォームにのみ適用され、target_type は 100エージェント用です。</li><li>'b' はバッチ/シェルスクリプトを表します。このスクリプトの'target_type' は常に 100エージェントです。このスクリプトは、 Windows ではバッチスクリプトとして、他のプラットフォームではシェルスクリプトとして動作します。</li><li>'s' は SQL スクリプトの略です。 'target_type' は常に 200サーバー、 300データベース、 400スキーマ、 500です。テーブル、 600 インデックス、 700 シーケンス、 800 ファンクション、 900 ビュー) または1000拡張子。</li></ul> |

次のページに続く

表 429 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| ➤ enabled | boolean | false | none | プローブがデフォルトで有効/無効かどうかを決定します。 |

次のページに続く

表 429 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| ➤ target_type | integer | false | none | プローブが適用されるオブジェクトのタイプを定義します。<ul><li>target_type 100のプローブは、エージェントの統計を収集します。</li><li>target_type 200のプローブは、収集を担当します。</li><li>target_type 300のプローブは、データベースの統計を収集します。</li><li>target_type 400<のプローブ/b>は、スキーマの統計を収集します。</li><li>target_type 500のプローブは、テーブルの統計を収集します。</li><li>次のプローブtarget_type 600は、インデックスの統計を収集します。</li><li>target_type 700のプローブは、シーケンスの統計を収集します。</li> <li>target_type 800のプローブは、ファンクションの統計を収集します。</li><li>target_type 900のプローブは、ファンクションの統計を収集します。view.</li><li>target_type 1000のプローブは、エクステンの統計を収集します。</li></ul> |

次のページに続く

表 429 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » » » » » any_server_version | boolean | false | none | このプローブは Postgres バリアントのバージョンに適用されるかどうかを判断します。詳細については、ドキュメントを参照してください。 |
| » » » » » any_extension_version | boolean | false | none | このプローブは拡張バリアントのバージョンに適用されるかどうかを判断します。詳細については、ドキュメントを参照してください。 |
| ➢ extension_name | string | false | none | 拡張機能の名前。<ul>有効な名前は次のとおりです。 edbspl 、 plpgsql 、 hstore 、 pldbgapi 、 edb_dblink_oci 、 edb_dblink_libpq 、 sslutils .</ul> |
| ➢ interval | integer | false | none | プローブが統計を収集する頻度を秒単位で定義します。10 以上である必要があります |
| ➢ lifetime | integer | false | none | 履歴データの保持期間を日単位で定義します。有効な範囲は 1〜365 です。 'discard_history' が true に設定されている場合、この値は効果がありません。 |
| ➢ discard_history | boolean | false | none | 履歴データを破棄するかどうかを定義します。'discard_history' が false に設定されている場合、このプローブの履歴データは保存されません。 |

次のページに続く

表 429 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| ➤ platform | string | false | none | このプローブを実行できるプラットフォームを定義します。この値は、エージェントレベルのプローブにのみ適用されます。 |
| ➤ probe_code | string | false | none | コードの定義ごとに統計を収集するためのSQL/Batch スクリプト/シェルスクリプト/WMI スクリプト。 |
| ➤ probe_columns | [object] | false | none | なし |
| ➤ pc_name | string | false | none | カラム名 |
| ➤ pc_data_type | string | false | none | 列のデータ型。サポートされているデータ型は次のとおりです。<ul><ul>bigint</ul><ul>char</ul><ul>10 進 数</ul><ul>倍 精 度</ul><ul>整 数</ul><ul>数 値</ul><ul>実数</ul><ul>テ キ ス ト</ul><ul>text[]</ul><ul>timestamp</ul><ul>t/ul> |
| ➤ pc_unit | string | false | none | 列の単位を返します。 |
| ➤ pc_col_type | string | false | none | 内部テーブルの主キーの一部として列を返すか、否か。 'k' はキー列を表し、'm' はメンバー/ノーマル列を表します。 |
| ➤ pc_graphable | boolean | false | none | 列のデータをチャートにグラフ化できるかどうかを決定します。 |
| ➤ pc_calc_pit | boolean | false | none | サーバーがこの列のメトリックデータのポイントインタイムを計算するかどうかを決定します。 |

次のページに続く

表 429 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| ➽ pc_pit_default | boolean | false | none | メトリックデータをデフォルトでポイントインタイムで保存するかどうかを決定します。詳細については、ドキュメントの'Custom Probes'セクションを参照してください。 |
| ➽ alternate_code | *Platform Compatibility* ] | false | none | なし |
| » server_version_id | integer | false | none | 監視対象サーバーのサーバーバージョン ID<table class="model"><thead><tr><th>server_version Server</th></tr></thead><thead /> <tbody><tr><td>10905</td><td>PostgreSQL 9.5</td></tr><tr><td>10906</td><td>PostgreS 9.6</td></tr ><tr><td>11000</td><td>PostgreSQL 10</td></tr><tr><td>11100</td><td>PostgreS 11</td></tr>< tr><td>11200</td><td>PostgreSQL 12</td></tr><tr><td>11300</td><td>PostgreS 13</td></tr><tr> <td>20905</td><td>Postgres Plus Advanced Server 9.5</td></tr><tr><td>20906</td><td>Postgres Plus Advanced Server 9.6</td></tr ><tr><td>21000</td><td> EDB Postgres Advanced Server 10</td></tr><tr><td>21100</td><td> EDB Postgres Advanced Server 11</td ></tr><tr><td>21200 </td><td> EDB Postgres Advanced Server 12</td></tr><tr><td>21300 </td><td> EDB Postgres Advanced Server 13</td></tr></tbody></table> |

次のページに続く

表 429 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » » » » » » server_probe_code | string | false | none | 指定されたサーバーバージョンで実行する SQL クエリー |
| » extension_version | integer | false | none | 拡張機能のバージョンを指定します。拡張ターゲットタイプに有効な |

*and*

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » _anonymous_ | object | false | none | なし |
| » internal_name | string | false | none | プローブがエクスポートされたシステムからのプローブの内部名 |

### 列挙値

| Property | Value |
|---|---|
| applicable_on_server | ALL |
| applicable_on_server | POSTGRES_SERVER |
| applicable_on_server | ADVANCED_SERVER |
| operator | > |
| operator | < |
| collection_method | w |
| collection_method | b |
| collection_method | s |
| platform | unix |
| platform | windows |
| pc_data_type | bigint |
| pc_data_type | char |
| pc_data_type | decimal |
| pc_data_type | double precision |
| pc_data_type | integer |
| pc_data_type | numeric |
| pc_data_type | real |
| pc_data_type | text |
| pc_data_type | text[] |
| pc_data_type | timestamp |
| pc_data_type | timestamptz |
| pc_col_type | m |

次のページに続く

表 431 – 前のページからの続き

| Property | Value |
|---|---|
| pc_col_type | k |

**注釈**

この操作には認証が必要ありません

### Import Custom Probes

#### post_probe_custom_import

コードサンプル

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v17/probe/custom/import/ \
  -H Content-Type: application/json \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Content-Type: application/json,
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.post(https://PEM-SERVER-IP/pem/api/v17/probe/custom/import/, headers =␣
↪headers)

print(r.json())
```

```
const inputBody = {
  "version": 1,
  "skip_overwrite": true,
  "probes": [
    {
      "probe_name": "string",
      "collection_method": "w",
      "enabled": true,
      "target_type": 0,
      "any_server_version": true,
      "any_extension_version": true,
```

(次のページに続く)

(前のページからの続き)

```
      "extension_name": "string",
      "interval": 0,
      "lifetime": 0,
      "discard_history": true,
      "platform": "unix",
      "probe_code": "string",
      "probe_columns": [
        {
          "pc_name": "string",
          "pc_data_type": "bigint",
          "pc_unit": "string",
          "pc_col_type": "m",
          "pc_graphable": true,
          "pc_calc_pit": true,
          "pc_pit_default": true
        }
      ],
      "alternate_code": [
        {
          "server_version_id": 0,
          "server_probe_code": "string",
          "extension_version": 0
        }
      ],
      "internal_name": "string"
    }
  ]
};
const headers = {
  Content-Type:application/json,
  Accept:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v17/probe/custom/import/,
{
  method: POST,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

POST /probe/custom/import/

エクスポートされたプローブを使用してカスタムプローブをインポートします。

**ボディパラメーター**

```
{
  "version": 1,
  "skip_overwrite": true,
  "probes": [
    {
      "probe_name": "string",
      "collection_method": "w",
      "enabled": true,
      "target_type": 0,
      "any_server_version": true,
      "any_extension_version": true,
      "extension_name": "string",
      "interval": 0,
      "lifetime": 0,
      "discard_history": true,
      "platform": "unix",
      "probe_code": "string",
      "probe_columns": [
        {
          "pc_name": "string",
          "pc_data_type": "bigint",
          "pc_unit": "string",
          "pc_col_type": "m",
          "pc_graphable": true,
          "pc_calc_pit": true,
          "pc_pit_default": true
        }
      ],
      "alternate_code": [
        {
          "server_version_id": 0,
          "server_probe_code": "string",
          "extension_version": 0
        }
      ],
      "internal_name": "string"
    }
  ]
}
```

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| X-Auth-Token | header | string | true | 承認トークン |
| body | body | *CustomProbeImportData* | true | なし |

### 応答例

```
[
  {
    "name": "string",
    "status": "string",
    "msg": "string"
  }
]
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ステータス 200 | Inline |

### レスポンススキーマ

ステータスコード **200**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » name | string | true | none | なし |
| » status | string | true | none | なし |
| » msg | string | true | none | なし |

**注釈**

この操作には認証が必要ありません

### Probe Configuration

#### get_probe_config_agent{agent_id}

コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/probe/config/agent/{agent_id} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/probe/config/agent/{agent_id},␣
↪headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch("https://PEM-SERVER-IP/pem/api/v17/probe/config/agent/{agent_id}", {
  method: "GET",

  headers: headers,
})
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /probe/config/agent/{agent_id}

すべてのエージェントレベルのプローブを取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| agent_id | path | integer | true | エージェント ID |
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例

```
[
  {
    "probe_id": 0,
    "probe_name": "string",
    "default_interval_min": 0,
    "interval_min": 0,
    "default_interval_sec": 0,
    "interval_sec": 0,
    "default_interval": 0,
    "interval": 0,
    "default_lifetime": 0,
    "lifetime": 0,
    "default_enabled": true,
    "enabled": true,
    "target_type_id_returned": 0
  }
]
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | Inline |

### レスポンススキーマ

ステータスコード **200**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| _anonymous_ | *Platform Compatibility* ] | false | none | なし |
| » probe_id | integer | false | none | プローブの一意の ID |

次のページに続く

表 437 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » probe_name | string | false | none | プローブの名前 |
| » default_interval_min | integer | false | none | プローブが統計を収集するデフォルトの間隔を分単位で定義します。指定できる値は0〜2880です。 |
| » interval_min | integer | false | none | プローブが統計を収集する頻度を分単位で定義します。指定できる値は 0〜2880 です。 |
| » default_interval_sec | integer | false | none | プローブが統計を収集するデフォルトの間隔を秒単位で定義します。指定できる値は 0〜59 です。 |
| » interval_sec | integer | false | none | プローブが統計を収集する頻度を秒単位で定義します。指定できる値は 0〜59 です。 |
| » default_interval | integer | false | none | プローブが統計を収集するデフォルトの間隔を秒単位で定義します。 |
| » interval | integer | false | none | プローブが統計を収集する頻度を秒単位で定義します。10 以上である必要があります |
| » default_lifetime | integer | false | none | デフォルトの履歴データの保持期間を日単位で定義します。 |
| » lifetime | integer | false | none | 履歴データの保持期間を日単位で定義します。有効な範囲は 1〜365 です。 'discard_history' が true に設定されている場合、この値は効果がありません。 |
| » default_enabled | boolean | false | none | デフォルトのプローブ有効/無効値を定義します。 |
| » enabled | boolean | false | none | プローブがデフォルトで有効/無効かどうかを指定します。 |

次のページに続く

表 437 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » target_type_id_returned | integer | false | none | 戻りターゲットタイプの値を定義します。<ul><li>target_type 100のプローブは、エージェントの統計を収集します。</li><li>target_type 200のプローブは、収集を担当します。</li><li>target_type 300のプローブは、データベースの統計を収集します。</li><li>target_type 400<のプローブ/b>は、スキーマの統計を収集します。</li><li>target_type 500のプローブは、テーブルの統計を収集します。</li><li>次のプローブtarget_type 600は、インデックスの統計を収集します。</li><li>target_type 700のプローブは、シーケンスの統計を収集します。</li> <li>target_type 800のプローブは、ファンクションの統計を収集します。</li><li>target_type 900のプローブは、ファンクションの統計を収集します。view.</li></ul> |

**注釈**

この操作には認証が必要ありません

### get_*probe_config_server*{server_id}

コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/probe/config/server/{server_id} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/probe/config/server/{server_id},␣
↪headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch("https://PEM-SERVER-IP/pem/api/v17/probe/config/server/{server_id}", {
  method: "GET",

  headers: headers,
})
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /probe/config/server/{server_id}

すべてのサーバーレベルのプローブを取得します。

#### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例

```
[
  {
    "probe_id": 0,
    "probe_name": "string",
    "default_interval_min": 0,
    "interval_min": 0,
    "default_interval_sec": 0,
    "interval_sec": 0,
    "default_interval": 0,
    "interval": 0,
    "default_lifetime": 0,
    "lifetime": 0,
    "default_enabled": true,
    "enabled": true,
    "target_type_id_returned": 0
  }
]
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | Inline |

### レスポンススキーマ

ステータスコード **200**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| _anonymous_ | *Platform Compatibility* ] | false | none | なし |
| » probe_id | integer | false | none | プローブの一意の ID |
| » probe_name | string | false | none | プローブの名前 |

次のページに続く

表 440 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » default_interval_min | integer | false | none | プローブが統計を収集するデフォルトの間隔を分単位で定義します。指定できる値は0〜2880です。 |
| » interval_min | integer | false | none | プローブが統計を収集する頻度を分単位で定義します。指定できる値は0〜2880 です。 |
| » default_interval_sec | integer | false | none | プローブが統計を収集するデフォルトの間隔を秒単位で定義します。指定できる値は 0〜59 です。 |
| » interval_sec | integer | false | none | プローブが統計を収集する頻度を秒単位で定義します。指定できる値は0〜59 です。 |
| » default_interval | integer | false | none | プローブが統計を収集するデフォルトの間隔を秒単位で定義します。 |
| » interval | integer | false | none | プローブが統計を収集する頻度を秒単位で定義します。10 以上である必要があります |
| » default_lifetime | integer | false | none | デフォルトの履歴データの保持期間を日単位で定義します。 |
| » lifetime | integer | false | none | 履歴データの保持期間を日単位で定義します。有効な範囲は 1〜365 です。 'discard_history' が true に設定されている場合、この値は効果がありません。 |
| » default_enabled | boolean | false | none | デフォルトのプローブ有効/無効値を定義します。 |
| » enabled | boolean | false | none | プローブがデフォルトで有効/無効かどうかを指定します。 |

次のページに続く

表 440 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » target_type_id_returned | integer | false | none | 戻りターゲットタイプの値を定義します。<ul><li>target_type 100のプローブは、エージェントの統計を収集します。</li><li>target_type 200のプローブは、収集を担当します。</li><li>target_type 300のプローブは、データベースの統計を収集します。</li><li>target_type 400<の プローブ/b>は、スキーマの統計を収集します。</li><li>target_type 500のプローブは、テーブルの統計を収集します。</li><li>次のプローブtarget_type 600は、インデックスの統計を収集します。</li><li>target_type 700のプローブは、シーケンスの統計を収集します。</li> <li>target_type 800のプローブは、ファンクションの統計を収集します。</li><li>target_type 900のプローブは、ファンクションの統計を収集します。view.</li></ul> |

**注釈**

この操作には認証が必要ありません

### get_*probe_config_server*{server_id}_{probe_id}

コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/probe/config/server/{server_id}/{probe_
↪id} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/probe/config/server/{server_id}/
↪{probe_id}, headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch(
  "https://PEM-SERVER-IP/pem/api/v17/probe/config/server/{server_id}/{probe_id}",
  {
    method: "GET",

    headers: headers,
  },
)
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /probe/config/server/{server_id}/{probe_id}

指定されたプローブ ID のサーバーレベルのプローブを取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| probe_id | path | integer | true | プローブ構成が返されるプローブ ID |
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例


```
{
  "probe_id": 0,
  "probe_name": "string",
  "default_interval_min": 0,
  "interval_min": 0,
  "default_interval_sec": 0,
  "interval_sec": 0,
  "default_interval": 0,
  "interval": 0,
  "default_lifetime": 0,
  "lifetime": 0,
  "default_enabled": true,
  "enabled": true,
  "target_type_id_returned": 0
}
```


### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | プローブ |

**注釈**

この操作には認証が必要ありません

### put_*probe_config_server*{server_id}_{probe_id}

コードサンプル

```
#  You can also use wget

curl -X PUT https://PEM-SERVER-IP/pem/api/v17/probe/config/server/{server_id}/{probe_
↪id} \
  -H Content-Type: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Content-Type: application/json,
  X-Auth-Token: string
}

r = requests.put(https://PEM-SERVER-IP/pem/api/v17/probe/config/server/{server_id}/
↪{probe_id}, headers = headers)

print(r.json())
```

```
const inputBody = {
  "interval_min": 0,
  "interval_sec": 0,
  "lifetime": 0,
  "enabled": true
};
const headers = {
  Content-Type:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v17/probe/config/server/{server_id}/{probe_id},
{
  method: PUT,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

```
PUT /probe/config/server/{server_id}/{probe_id}
```

指定されたプローブ ID のサーバーレベルのプローブ構成を更新します。

**ボディパラメーター**

```
{
  "interval_min": 0,
  "interval_sec": 0,
  "lifetime": 0,
  "enabled": true
}
```

**パラメーター**

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| probe_id | path | integer | true | プローブ構成が返されるプローブ ID |
| X-Auth-Token | header | string | true | 承認トークン |
| body | body | *ProbeConfigData* | true | なし |

**応答例**

**応答**

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ステータス 200 | None |

**レスポンススキーマ**

**注釈**

この操作には認証が必要ありません

### get_*probe_config_server*{server_id}*database*{database_name}

**コードサンプル**

```
#  You can also use wget
```

(次のページに続く)

(前のページからの続き)

```
curl -X GET https://PEM-SERVER-IP/pem/api/v17/probe/config/server/{server_id}/
↪database/{database_name} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/probe/config/server/{server_id}/
↪database/{database_name}, headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch(
  "https://PEM-SERVER-IP/pem/api/v17/probe/config/server/{server_id}/database/
↪{database_name}",
  {
    method: "GET",

    headers: headers,
  },
)
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /probe/config/server/{server_id}/database/{database_name}

すべてのデータベースレベルのプローブを取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| database_name | path | string | true | データベース名 |
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例

```
[
  {
    "probe_id": 0,
    "probe_name": "string",
    "default_interval_min": 0,
    "interval_min": 0,
    "default_interval_sec": 0,
    "interval_sec": 0,
    "default_interval": 0,
    "interval": 0,
    "default_lifetime": 0,
    "lifetime": 0,
    "default_enabled": true,
    "enabled": true,
    "target_type_id_returned": 0
  }
]
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | Inline |

### レスポンススキーマ

ステータスコード **200**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| _anonymous_ | *Platform Compatibility* ] | false | none | なし |

次のページに続く

表 447 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » probe_id | integer | false | none | プローブの一意の ID |
| » probe_name | string | false | none | プローブの名前 |
| » default_interval_min | integer | false | none | プローブが統計を収集するデフォルトの間隔を分単位で定義します。指定できる値は 0〜2880 です。 |
| » interval_min | integer | false | none | プローブが統計を収集する頻度を分単位で定義します。指定できる値は 0〜2880 です。 |
| » default_interval_sec | integer | false | none | プローブが統計を収集するデフォルトの間隔を秒単位で定義します。指定できる値は 0〜59 です。 |
| » interval_sec | integer | false | none | プローブが統計を収集する頻度を秒単位で定義します。指定できる値は 0〜59 です。 |
| » default_interval | integer | false | none | プローブが統計を収集するデフォルトの間隔を秒単位で定義します。 |
| » interval | integer | false | none | プローブが統計を収集する頻度を秒単位で定義します。10 以上である必要があります |
| » default_lifetime | integer | false | none | デフォルトの履歴データの保持期間を日単位で定義します。 |
| » lifetime | integer | false | none | 履歴データの保持期間を日単位で定義します。有効な範囲は 1〜365 です。 'discard_history' が true に設定されている場合、この値は効果がありません。 |
| » default_enabled | boolean | false | none | デフォルトのプローブ有効/無効値を定義します。 |
| » enabled | boolean | false | none | プローブがデフォルトで有効/無効かどうかを指定します。 |

次のページに続く

表 447 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » target_type_id_returned | integer | false | none | 戻りターゲットタイプの値を定義します。<ul><li>target_type 100のプローブは、エージェントの統計を収集します。</li><li>target_type 200のプローブは、収集を担当します。</li><li>target_type 300のプローブは、データベースの統計を収集します。</li><li>target_type 400<のプローブ/b>は、スキーマの統計を収集します。</li><li>target_type 500のプローブは、テーブルの統計を収集します。</li><li>次のプローブtarget_type 600は、インデックスの統計を収集します。</li><li>target_type 700のプローブは、シーケンスの統計を収集します。</li> <li>target_type 800のプローブは、ファンクションの統計を収集します。</li><li>target_type 900のプローブは、ファンクションの統計を収集します。view.</li></ul> |

**注釈**

この操作には認証が必要ありません

### get_*probe_config_server*{server_id}*database*{database_name}_{probe_id}

コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/probe/config/server/{server_id}/
↪database/{database_name}/{probe_id} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/probe/config/server/{server_id}/
↪database/{database_name}/{probe_id}, headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch(
  "https://PEM-SERVER-IP/pem/api/v17/probe/config/server/{server_id}/database/
↪{database_name}/{probe_id}",
  {
    method: "GET",

    headers: headers,
  },
)
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /probe/config/server/{server_id}/database/{database_name}/{probe_id}

指定されたプローブ ID のデータベースレベルのプローブを取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| database_name | path | string | true | データベース名 |
| probe_id | path | integer | true | プローブ構成が返されるプローブ ID。 |
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例


```
{
  "probe_id": 0,
  "probe_name": "string",
  "default_interval_min": 0,
  "interval_min": 0,
  "default_interval_sec": 0,
  "interval_sec": 0,
  "default_interval": 0,
  "interval": 0,
  "default_lifetime": 0,
  "lifetime": 0,
  "default_enabled": true,
  "enabled": true,
  "target_type_id_returned": 0
}
```


### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | プローブ |

**注釈**

この操作には認証が必要ありません

### put_*probe_config_server*{server_id}*database*{database_name}_{probe_id}

コードサンプル

```
#  You can also use wget

curl -X PUT https://PEM-SERVER-IP/pem/api/v17/probe/config/server/{server_id}/
↪database/{database_name}/{probe_id} \
  -H Content-Type: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Content-Type: application/json,
  X-Auth-Token: string
}

r = requests.put(https://PEM-SERVER-IP/pem/api/v17/probe/config/server/{server_id}/
↪database/{database_name}/{probe_id}, headers = headers)

print(r.json())
```

```
const inputBody = {
  "interval_min": 0,
  "interval_sec": 0,
  "lifetime": 0,
  "enabled": true
};
const headers = {
  Content-Type:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v17/probe/config/server/{server_id}/database/
↪{database_name}/{probe_id},
{
  method: PUT,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

```
PUT /probe/config/server/{server_id}/database/{database_name}/{probe_id}
```

指定されたプローブ ID のデータベースレベルのプローブ構成を更新します。

**ボディパラメーター**

```
{
  "interval_min": 0,
  "interval_sec": 0,
  "lifetime": 0,
  "enabled": true
}
```

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| database_name | path | string | true | データベース名 |
| probe_id | path | integer | true | プローブ構成が返されるプローブ ID |
| X-Auth-Token | header | string | true | 承認トークン |
| body | body | *ProbeConfigData* | true | なし |

**応答例**

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ステータス 200 | None |

### レスポンススキーマ

> **注釈**
>
> この操作には認証が必要ありません

## get_*probe_config_server*{server_id}*database*{database_name}*schema*{schema_name}

**コードサンプル**

```
#  You can also use wget
```

(次のページに続く)

(前のページからの続き)

```
curl -X GET https://PEM-SERVER-IP/pem/api/v17/probe/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/probe/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}, headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch(
  "https://PEM-SERVER-IP/pem/api/v17/probe/config/server/{server_id}/database/
↪{database_name}/schema/{schema_name}",
  {
    method: "GET",

    headers: headers,
  },
)
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /probe/config/server/{server_id}/database/{database_name}/schema/{schema_name}

すべてのスキーマレベルのプローブを取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| database_name | path | string | true | データベース名 |
| schema_name | path | string | true | スキーマ名 |
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例

```
[
  {
    "probe_id": 0,
    "probe_name": "string",
    "default_interval_min": 0,
    "interval_min": 0,
    "default_interval_sec": 0,
    "interval_sec": 0,
    "default_interval": 0,
    "interval": 0,
    "default_lifetime": 0,
    "lifetime": 0,
    "default_enabled": true,
    "enabled": true,
    "target_type_id_returned": 0
  }
]
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | Inline |

### レスポンススキーマ

ステータスコード **200**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| _anonymous_ | *Platform Compatibility* ] | false | none | なし |
| » probe_id | integer | false | none | プローブの一意の ID |
| » probe_name | string | false | none | プローブの名前 |
| » default_interval_min | integer | false | none | プローブが統計を収集するデフォルトの間隔を分単位で定義します。指定できる値は 0〜2880 です。 |
| » interval_min | integer | false | none | プローブが統計を収集する頻度を分単位で定義します。指定できる値は 0〜2880 です。 |
| » default_interval_sec | integer | false | none | プローブが統計を収集するデフォルトの間隔を秒単位で定義します。指定できる値は 0〜59 です。 |
| » interval_sec | integer | false | none | プローブが統計を収集する頻度を秒単位で定義します。指定できる値は 0〜59 です。 |
| » default_interval | integer | false | none | プローブが統計を収集するデフォルトの間隔を秒単位で定義します。 |
| » interval | integer | false | none | プローブが統計を収集する頻度を秒単位で定義します。10 以上である必要があります |
| » default_lifetime | integer | false | none | デフォルトの履歴データの保持期間を日単位で定義します。 |
| » lifetime | integer | false | none | 履歴データの保持期間を日単位で定義します。有効な範囲は 1〜365 です。 'discard_history' が true に設定されている場合、この値は効果がありません。 |
| » default_enabled | boolean | false | none | デフォルトのプローブ有効/無効値を定義します。 |
| » enabled | boolean | false | none | プローブがデフォルトで有効/無効かどうかを指定します。 |

次のページに続く

表 454 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » target_type_id_returned | integer | false | none | 戻りターゲットタイプの値を定義します。<ul><li>target_type 100のプローブは、エージェントの統計を収集します。</li><li>target_type 200のプローブは、収集を担当します。</li><li>target_type 300のプローブは、データベースの統計を収集します。</li><li>target_type 400<のプローブ/b>は、スキーマの統計を収集します。</li><li>target_type 500のプローブは、テーブルの統計を収集します。</li><li>次のプローブtarget_type 600は、インデックスの統計を収集します。</li><li>target_type 700のプローブは、シーケンスの統計を収集します。</li> <li>target_type 800のプローブは、ファンクションの統計を収集します。</li><li>target_type 900のプローブは、ファンクションの統計を収集します。view.</li></ul> |

**注釈**

この操作には認証が必要ありません

### get_*probe_config_server*{server_id}*database*{database_name}*schema*{schema_name}_{probe_id}

コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/probe/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/{probe_id} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/probe/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/{probe_id}, headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch(
  "https://PEM-SERVER-IP/pem/api/v17/probe/config/server/{server_id}/database/
↪{database_name}/schema/{schema_name}/{probe_id}",
  {
    method: "GET",

    headers: headers,
  },
)
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

```
GET /probe/config/server/{server_id}/database/{database_name}/schema/{schema_name}/
{probe_id}
```

指定されたプローブ ID のスキーマレベルのプローブを取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| database_name | path | string | true | データベース名 |
| schema_name | path | string | true | スキーマ名 |
| probe_id | path | integer | true | プローブ構成が返されるプローブ ID |
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例


```
{
  "probe_id": 0,
  "probe_name": "string",
  "default_interval_min": 0,
  "interval_min": 0,
  "default_interval_sec": 0,
  "interval_sec": 0,
  "default_interval": 0,
  "interval": 0,
  "default_lifetime": 0,
  "lifetime": 0,
  "default_enabled": true,
  "enabled": true,
  "target_type_id_returned": 0
}
```


### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | プローブ |

**注釈**

この操作には認証が必要ありません

### put_*probe_config_server*{server_id}*database*{database_name}*schema*{schema_name}_{probe_id}

コードサンプル

```
#  You can also use wget

curl -X PUT https://PEM-SERVER-IP/pem/api/v17/probe/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/{probe_id} \
  -H Content-Type: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Content-Type: application/json,
  X-Auth-Token: string
}

r = requests.put(https://PEM-SERVER-IP/pem/api/v17/probe/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/{probe_id}, headers = headers)

print(r.json())
```

```
const inputBody = {
  "interval_min": 0,
  "interval_sec": 0,
  "lifetime": 0,
  "enabled": true
};
const headers = {
  Content-Type:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v17/probe/config/server/{server_id}/database/
↪{database_name}/schema/{schema_name}/{probe_id},
{
  method: PUT,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

```
PUT /probe/config/server/{server_id}/database/{database_name}/schema/{schema_name}/
```

{probe_id}

指定されたプローブ ID のスキーマレベルのプローブ構成を更新します。

**ボディパラメーター**

```
{
  "interval_min": 0,
  "interval_sec": 0,
  "lifetime": 0,
  "enabled": true
}
```

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| database_name | path | string | true | データベース名 |
| schema_name | path | string | true | スキーマ名 |
| probe_id | path | integer | true | プローブ構成が返されるプローブ ID |
| X-Auth-Token | header | string | true | 承認トークン |
| body | body | *ProbeConfigData* | true | なし |

**応答例**

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ステータス 200 | None |

### レスポンススキーマ

**注釈**

この操作には認証が必要ありません

**get_*probe_config_server*{server_id}*database*{database_name}*schema*{schema_name}*table*{table_name}**

**コードサンプル**

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/probe/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/table/{table_name} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/probe/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/table/{table_name}, headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch(
  "https://PEM-SERVER-IP/pem/api/v17/probe/config/server/{server_id}/database/
↪{database_name}/schema/{schema_name}/table/{table_name}",
  {
    method: "GET",

    headers: headers,
  },
)
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /probe/config/server/{server_id}/database/{database_name}/schema/{schema_name}/table/{table_name}

すべてのテーブルレベルのプローブを取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| database_name | path | string | true | データベース名 |
| schema_name | path | string | true | スキーマ名 |
| table_name | path | string | true | テーブル名 |
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例

```
[
  {
    "probe_id": 0,
    "probe_name": "string",
    "default_interval_min": 0,
    "interval_min": 0,
    "default_interval_sec": 0,
    "interval_sec": 0,
    "default_interval": 0,
    "interval": 0,
    "default_lifetime": 0,
    "lifetime": 0,
    "default_enabled": true,
    "enabled": true,
    "target_type_id_returned": 0
  }
]
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | Inline |

### レスポンススキーマ

ステータスコード **200**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| _anonymous_ | *Platform Compatibility* ] | false | none | なし |
| » probe_id | integer | false | none | プローブの一意の ID |
| » probe_name | string | false | none | プローブの名前 |
| » default_interval_min | integer | false | none | プローブが統計を収集するデフォルトの間隔を分単位で定義します。指定できる値は 0〜2880 です。 |
| » interval_min | integer | false | none | プローブが統計を収集する頻度を分単位で定義します。指定できる値は 0〜2880 です。 |
| » default_interval_sec | integer | false | none | プローブが統計を収集するデフォルトの間隔を秒単位で定義します。指定できる値は 0〜59 です。 |
| » interval_sec | integer | false | none | プローブが統計を収集する頻度を秒単位で定義します。指定できる値は 0〜59 です。 |
| » default_interval | integer | false | none | プローブが統計を収集するデフォルトの間隔を秒単位で定義します。 |
| » interval | integer | false | none | プローブが統計を収集する頻度を秒単位で定義します。10 以上である必要があります |
| » default_lifetime | integer | false | none | デフォルトの履歴データの保持期間を日単位で定義します。 |
| » lifetime | integer | false | none | 履歴データの保持期間を日単位で定義します。有効な範囲は 1〜365 です。 'discard_history' が true に設定されている場合、この値は効果がありません。 |
| » default_enabled | boolean | false | none | デフォルトのプローブ有効/無効値を定義します。 |
| » enabled | boolean | false | none | プローブがデフォルトで有効/無効かどうかを指定します。 |

次のページに続く

表 461 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » target_type_id_returned | integer | false | none | 戻りターゲットタイプの値を定義します。<ul><li>target_type 100のプローブは、エージェントの統計を収集します。</li><li>target_type 200のプローブは、収集を担当します。</li><li>target_type 300のプローブは、データベースの統計を収集します。</li><li>target_type 400<のプローブ/b>は、スキーマの統計を収集します。</li><li>target_type 500のプローブは、テーブルの統計を収集します。</li><li>次のプローブtarget_type 600は、インデックスの統計を収集します。</li><li>target_type 700のプローブは、シーケンスの統計を収集します。</li> <li>target_type 800のプローブは、ファンクションの統計を収集します。</li><li>target_type 900のプローブは、ファンクションの統計を収集します。view.</li></ul> |

**注釈**

この操作には認証が必要ありません

### get_*probe_config_server*{server_id}*database*{database_name}*schema*{schema_name}*table*{table_name}_{probe

コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/probe/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/table/{table_name}/{probe_id} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/probe/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/table/{table_name}/{probe_id},␣
↪headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch(
  "https://PEM-SERVER-IP/pem/api/v17/probe/config/server/{server_id}/database/
↪{database_name}/schema/{schema_name}/table/{table_name}/{probe_id}",
  {
    method: "GET",

    headers: headers,
  },
)
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

```
GET /probe/config/server/{server_id}/database/{database_name}/schema/{schema_name}/
table/{table_name}/{probe_id}
```

指定されたプローブ ID のテーブルレベルのプローブを取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| database_name | path | string | true | データベース名 |
| schema_name | path | string | true | スキーマ名 |
| table_name | path | string | true | テーブル名 |
| probe_id | path | integer | true | プローブ構成が返されるプローブ ID |
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例

```
{
  "probe_id": 0,
  "probe_name": "string",
  "default_interval_min": 0,
  "interval_min": 0,
  "default_interval_sec": 0,
  "interval_sec": 0,
  "default_interval": 0,
  "interval": 0,
  "default_lifetime": 0,
  "lifetime": 0,
  "default_enabled": true,
  "enabled": true,
  "target_type_id_returned": 0
}
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | プローブ |

**注釈**

この操作には認証が必要ありません

### get_*probe_config_server*{server_id}*database*{database_name}*schema*{schema_name}*index*{index_name}

コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/probe/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/index/{index_name} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/probe/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/index/{index_name}, headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch(
  "https://PEM-SERVER-IP/pem/api/v17/probe/config/server/{server_id}/database/
↪{database_name}/schema/{schema_name}/index/{index_name}",
  {
    method: "GET",

    headers: headers,
  },
)
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /probe/config/server/{server_id}/database/{database_name}/schema/{schema_name}/index/{index_name}

すべてのインデックスレベルのプローブを取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| database_name | path | string | true | データベース名 |
| schema_name | path | string | true | スキーマ名 |
| index_name | path | string | true | インデックス名 |
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例

```
[
  {
    "probe_id": 0,
    "probe_name": "string",
    "default_interval_min": 0,
    "interval_min": 0,
    "default_interval_sec": 0,
    "interval_sec": 0,
    "default_interval": 0,
    "interval": 0,
    "default_lifetime": 0,
    "lifetime": 0,
    "default_enabled": true,
    "enabled": true,
    "target_type_id_returned": 0
  }
]
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | Inline |

### レスポンススキーマ

ステータスコード **200**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| _anonymous_ | *Platform Compatibility* ] | false | none | なし |
| » probe_id | integer | false | none | プローブの一意の ID |
| » probe_name | string | false | none | プローブの名前 |
| » default_interval_min | integer | false | none | プローブが統計を収集するデフォルトの間隔を分単位で定義します。指定できる値は 0〜2880 です。 |
| » interval_min | integer | false | none | プローブが統計を収集する頻度を分単位で定義します。指定できる値は 0〜2880 です。 |
| » default_interval_sec | integer | false | none | プローブが統計を収集するデフォルトの間隔を秒単位で定義します。指定できる値は 0〜59 です。 |
| » interval_sec | integer | false | none | プローブが統計を収集する頻度を秒単位で定義します。指定できる値は 0〜59 です。 |
| » default_interval | integer | false | none | プローブが統計を収集するデフォルトの間隔を秒単位で定義します。 |
| » interval | integer | false | none | プローブが統計を収集する頻度を秒単位で定義します。10 以上である必要があります |
| » default_lifetime | integer | false | none | デフォルトの履歴データの保持期間を日単位で定義します。 |
| » lifetime | integer | false | none | 履歴データの保持期間を日単位で定義します。有効な範囲は 1〜365 です。 'discard_history' が true に設定されている場合、この値は効果がありません。 |
| » default_enabled | boolean | false | none | デフォルトのプローブ有効/無効値を定義します。 |
| » enabled | boolean | false | none | プローブがデフォルトで有効/無効かどうかを指定します。 |

次のページに続く

表 466 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » target_type_id_returned | integer | false | none | 戻りターゲットタイプの値を定義します。<ul><li>target_type 100のプローブは、エージェントの統計を収集します。</li><li>target_type 200のプローブは、収集を担当します。</li><li>target_type 300のプローブは、データベースの統計を収集します。</li><li>target_type 400<の プローブ/b>は、スキーマの統計を収集します。</li><li>target_type 500のプローブは、テーブルの統計を収集します。</li><li>次のプローブ target_type 600は、インデックスの統計を収集します。</li><li>target_type 700のプローブは、シーケンスの統計を収集します。</li> <li>target_type 800のプローブは、ファンクションの統計を収集します。</li><li>target_type 900のプローブは、ファンクションの統計を収集します。view.</li></ul> |

**注釈**

この操作には認証が必要ありません

### get_*probe_config_server*{server_id}*database*{database_name}*schema*{schema_name}*index*{index_name}_{prob

コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/probe/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/index/{index_name}/{probe_id} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/probe/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/index/{index_name}/{probe_id},␣
↪headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch(
  "https://PEM-SERVER-IP/pem/api/v17/probe/config/server/{server_id}/database/
↪{database_name}/schema/{schema_name}/index/{index_name}/{probe_id}",
  {
    method: "GET",

    headers: headers,
  },
)
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

```
GET /probe/config/server/{server_id}/database/{database_name}/schema/{schema_name}/
index/{index_name}/{probe_id}
```

指定されたプローブ ID のインデックスレベルプローブを取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| database_name | path | string | true | データベース名 |
| schema_name | path | string | true | スキーマ名 |
| index_name | path | string | true | インデックス名 |
| probe_id | path | integer | true | プローブ構成が返されるプローブ ID |
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例

```
{
  "probe_id": 0,
  "probe_name": "string",
  "default_interval_min": 0,
  "interval_min": 0,
  "default_interval_sec": 0,
  "interval_sec": 0,
  "default_interval": 0,
  "interval": 0,
  "default_lifetime": 0,
  "lifetime": 0,
  "default_enabled": true,
  "enabled": true,
  "target_type_id_returned": 0
}
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | プローブ |

**注釈**

この操作には認証が必要ありません

**get_*probe_config_server*{server_id}*database*{database_name}*schema*{schema_name}*sequence*{sequence_nam**

コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/probe/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/sequence/{sequence_name} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/probe/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/sequence/{sequence_name}, headers =␣
↪headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch(
  "https://PEM-SERVER-IP/pem/api/v17/probe/config/server/{server_id}/database/
↪{database_name}/schema/{schema_name}/sequence/{sequence_name}",
  {
    method: "GET",

    headers: headers,
  },
)
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /probe/config/server/{server_id}/database/{database_name}/schema/{schema_name}/sequence/{sequence_name}

すべてのシーケンスレベルのプローブを取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| database_name | path | string | true | データベース名 |
| schema_name | path | string | true | スキーマ名 |
| sequence_name | path | string | true | シークエンス名 |
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例

```
[
  {
    "probe_id": 0,
    "probe_name": "string",
    "default_interval_min": 0,
    "interval_min": 0,
    "default_interval_sec": 0,
    "interval_sec": 0,
    "default_interval": 0,
    "interval": 0,
    "default_lifetime": 0,
    "lifetime": 0,
    "default_enabled": true,
    "enabled": true,
    "target_type_id_returned": 0
  }
]
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | Inline |

### レスポンススキーマ

ステータスコード **200**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| _anonymous_ | *Platform Compatibility* ] | false | none | なし |
| » probe_id | integer | false | none | プローブの一意の ID |
| » probe_name | string | false | none | プローブの名前 |
| » default_interval_min | integer | false | none | プローブが統計を収集するデフォルトの間隔を分単位で定義します。指定できる値は 0〜2880 です。 |
| » interval_min | integer | false | none | プローブが統計を収集する頻度を分単位で定義します。指定できる値は 0〜2880 です。 |
| » default_interval_sec | integer | false | none | プローブが統計を収集するデフォルトの間隔を秒単位で定義します。指定できる値は 0〜59 です。 |
| » interval_sec | integer | false | none | プローブが統計を収集する頻度を秒単位で定義します。指定できる値は 0〜59 です。 |
| » default_interval | integer | false | none | プローブが統計を収集するデフォルトの間隔を秒単位で定義します。 |
| » interval | integer | false | none | プローブが統計を収集する頻度を秒単位で定義します。10 以上である必要があります |
| » default_lifetime | integer | false | none | デフォルトの履歴データの保持期間を日単位で定義します。 |
| » lifetime | integer | false | none | 履歴データの保持期間を日単位で定義します。有効な範囲は 1〜365 です。 'discard_history' が true に設定されている場合、この値は効果がありません。 |

次のページに続く

表 471 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » default_enabled | boolean | false | none | デフォルトのプローブ有効/無効値を定義します。 |
| » enabled | boolean | false | none | プローブがデフォルトで有効/無効かどうかを指定します。 |
| » target_type_id_returned | integer | false | none | 戻りターゲットタイプの値を定義します。<ul><li>target_type 100のプローブは、エージェントの統計を収集します。</li><li>target_type 200のプローブは、収集を担当します。</li><li>target_type 300のプローブは、データベースの統計を収集します。</li><li>target_type 400<の プローブ/b>は、スキーマの統計を収集します。</li><li>target_type 500のプローブは、テーブルの統計を収集します。</li><li>次のプローブtarget_type 600は、インデックスの統計を収集します。</li><li>target_type 700のプローブは、シーケンスの統計を収集します。</li> <li>target_type 800のプローブは、ファンクションの統計を収集します。</li><li>target_type 900のプローブは、ファンクションの統計を収集します。view.</li></ul> |

> **注釈**
>
> この操作には認証が必要ありません

### get_*probe_config_server*{server_id}*database*{database_name}*schema*{schema_name}*sequence*{sequence_nam

#### コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/probe/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/sequence/{sequence_name}/{probe_id} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/probe/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/sequence/{sequence_name}/{probe_id},␣
↪headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch(
  "https://PEM-SERVER-IP/pem/api/v17/probe/config/server/{server_id}/database/
↪{database_name}/schema/{schema_name}/sequence/{sequence_name}/{probe_id}",
  {
    method: "GET",

    headers: headers,
  },
)
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
```

(次のページに続く)

(前のページからの続き)

```
  console.log(body);
});
```

`GET /probe/config/server/{server_id}/database/{database_name}/schema/{schema_name}/sequence/{sequence_name}/{probe_id}`

指定されたプローブ ID の配列レベルのプローブを取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| database_name | path | string | true | データベース名 |
| schema_name | path | string | true | スキーマ名 |
| sequence_name | path | string | true | シークエンス名 |
| probe_id | path | integer | true | プローブ構成が返されるプローブ ID |
| X-Auth-Token | header | string | true | 承認トークン |

**応答例**

```
{
  "probe_id": 0,
  "probe_name": "string",
  "default_interval_min": 0,
  "interval_min": 0,
  "default_interval_sec": 0,
  "interval_sec": 0,
  "default_interval": 0,
  "interval": 0,
  "default_lifetime": 0,
  "lifetime": 0,
  "default_enabled": true,
  "enabled": true,
  "target_type_id_returned": 0
}
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | プローブ |

**注釈**

この操作には認証が必要ありません

**get_*probe_config_server*{server_id}*database*{database_name}*schema*{schema_name}*function*{function_name}_**

コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/probe/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/function/{function_name}/ \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/probe/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/function/{function_name}/, headers =␣
↪headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch(
  "https://PEM-SERVER-IP/pem/api/v17/probe/config/server/{server_id}/database/
↪{database_name}/schema/{schema_name}/function/{function_name}/",
  {
    method: "GET",
```

(次のページに続く)

(前のページからの続き)

```
    headers: headers,
  },
)
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /probe/config/server/{server_id}/database/{database_name}/schema/{schema_name}/function/{function_name}/

すべてのファンクションレベルのプローブを取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| database_name | path | string | true | データベース名 |
| schema_name | path | string | true | スキーマ名 |
| function_name | path | string | true | ファンクション名 |
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例

```
[
  {
    "probe_id": 0,
    "probe_name": "string",
    "default_interval_min": 0,
    "interval_min": 0,
    "default_interval_sec": 0,
    "interval_sec": 0,
    "default_interval": 0,
    "interval": 0,
    "default_lifetime": 0,
    "lifetime": 0,
    "default_enabled": true,
    "enabled": true,
    "target_type_id_returned": 0
  }
]
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | Inline |

### レスポンススキーマ

ステータスコード **200**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| _anonymous_ | *Platform Compatibility* ] | false | none | なし |
| » probe_id | integer | false | none | プローブの一意の ID |
| » probe_name | string | false | none | プローブの名前 |
| » default_interval_min | integer | false | none | プローブが統計を収集するデフォルトの間隔を分単位で定義します。指定できる値は 0〜2880 です。 |
| » interval_min | integer | false | none | プローブが統計を収集する頻度を分単位で定義します。指定できる値は 0〜2880 です。 |
| » default_interval_sec | integer | false | none | プローブが統計を収集するデフォルトの間隔を秒単位で定義します。指定できる値は 0〜59 です。 |
| » interval_sec | integer | false | none | プローブが統計を収集する頻度を秒単位で定義します。指定できる値は 0〜59 です。 |
| » default_interval | integer | false | none | プローブが統計を収集するデフォルトの間隔を秒単位で定義します。 |
| » interval | integer | false | none | プローブが統計を収集する頻度を秒単位で定義します。10 以上である必要があります |

次のページに続く

表 476 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » default_lifetime | integer | false | none | デフォルトの履歴データの保持期間を日単位で定義します。 |
| » lifetime | integer | false | none | 履歴データの保持期間を日単位で定義します。有効な範囲は 1〜365 です。 'discard_history' が true に設定されている場合、この値は効果がありません。 |
| » default_enabled | boolean | false | none | デフォルトのプローブ有効/無効値を定義します。 |
| » enabled | boolean | false | none | プローブがデフォルトで有効/無効かどうかを指定します。 |

次のページに続く

表 476 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » target_type_id_returned | integer | false | none | 戻りターゲットタイプの値を定義します。<ul><li>target_type 100のプローブは、エージェントの統計を収集します。</li><li>target_type 200のプローブは、収集を担当します。</li><li>target_type 300のプローブは、データベースの統計を収集します。</li><li>target_type 400<のプローブ/b>は、スキーマの統計を収集します。</li><li>target_type 500のプローブは、テーブルの統計を収集します。</li><li>次のプローブtarget_type 600は、インデックスの統計を収集します。</li><li>target_type 700のプローブは、シーケンスの統計を収集します。</li> <li>target_type 800のプローブは、ファンクションの統計を収集します。</li><li>target_type 900のプローブは、ファンクションの統計を収集します。view.</li></ul> |

**注釈**

この操作には認証が必要ありません

### get_*probe_config_server*{server_id}*database*{database_name}*schema*{schema_name}*function*{function_name}_

コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/probe/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/function/{function_name}/{probe_id} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/probe/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/function/{function_name}/{probe_id},␣
↪headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch(
  "https://PEM-SERVER-IP/pem/api/v17/probe/config/server/{server_id}/database/
↪{database_name}/schema/{schema_name}/function/{function_name}/{probe_id}",
  {
    method: "GET",

    headers: headers,
  },
)
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

```
GET /probe/config/server/{server_id}/database/{database_name}/schema/{schema_name}/
function/{function_name}/{probe_id}
```

指定されたプローブ ID のファンクションレベルのプローブを取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| database_name | path | string | true | データベース名 |
| schema_name | path | string | true | スキーマ名 |
| function_name | path | string | true | ファンクション名 |
| probe_id | path | integer | true | プローブ構成が返されるプローブ ID |
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例

```
{
  "probe_id": 0,
  "probe_name": "string",
  "default_interval_min": 0,
  "interval_min": 0,
  "default_interval_sec": 0,
  "interval_sec": 0,
  "default_interval": 0,
  "interval": 0,
  "default_lifetime": 0,
  "lifetime": 0,
  "default_enabled": true,
  "enabled": true,
  "target_type_id_returned": 0
}
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | プローブ |

**注釈**

この操作には認証が必要ありません

### get_*probe_config_server*{server_id}*database*{database_name}*schema*{schema_name}*view*{view_name}

コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/probe/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/view/{view_name} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/probe/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/view/{view_name}, headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch(
  "https://PEM-SERVER-IP/pem/api/v17/probe/config/server/{server_id}/database/
↪{database_name}/schema/{schema_name}/view/{view_name}",
  {
    method: "GET",

    headers: headers,
  },
)
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

```
GET /probe/config/server/{server_id}/database/{database_name}/schema/{schema_name}/
view/{view_name}
```

すべてのビューレベルのプローブを取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| database_name | path | string | true | データベース名 |
| schema_name | path | string | true | スキーマ名 |
| view_name | path | string | true | ビュー名 |
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例

```
[
  {
    "probe_id": 0,
    "probe_name": "string",
    "default_interval_min": 0,
    "interval_min": 0,
    "default_interval_sec": 0,
    "interval_sec": 0,
    "default_interval": 0,
    "interval": 0,
    "default_lifetime": 0,
    "lifetime": 0,
    "default_enabled": true,
    "enabled": true,
    "target_type_id_returned": 0
  }
]
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | Inline |

### レスポンススキーマ

ステータスコード **200**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| _anonymous_ | *Platform Compatibility* ] | false | none | なし |
| » probe_id | integer | false | none | プローブの一意の ID |
| » probe_name | string | false | none | プローブの名前 |
| » default_interval_min | integer | false | none | プローブが統計を収集するデフォルトの間隔を分単位で定義します。指定できる値は 0〜2880 です。 |
| » interval_min | integer | false | none | プローブが統計を収集する頻度を分単位で定義します。指定できる値は 0〜2880 です。 |
| » default_interval_sec | integer | false | none | プローブが統計を収集するデフォルトの間隔を秒単位で定義します。指定できる値は 0〜59 です。 |
| » interval_sec | integer | false | none | プローブが統計を収集する頻度を秒単位で定義します。指定できる値は 0〜59 です。 |
| » default_interval | integer | false | none | プローブが統計を収集するデフォルトの間隔を秒単位で定義します。 |
| » interval | integer | false | none | プローブが統計を収集する頻度を秒単位で定義します。10 以上である必要があります |
| » default_lifetime | integer | false | none | デフォルトの履歴データの保持期間を日単位で定義します。 |
| » lifetime | integer | false | none | 履歴データの保持期間を日単位で定義します。有効な範囲は 1〜365 です。 'discard_history' が true に設定されている場合、この値は効果がありません。 |
| » default_enabled | boolean | false | none | デフォルトのプローブ有効/無効値を定義します。 |
| » enabled | boolean | false | none | プローブがデフォルトで有効/無効かどうかを指定します。 |

次のページに続く

表 481 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » target_type_id_returned | integer | false | none | 戻りターゲットタイプの値を定義します。<ul><li>target_type 100のプローブは、エージェントの統計を収集します。</li><li>target_type 200のプローブは、収集を担当します。</li><li>target_type 300のプローブは、データベースの統計を収集します。</li><li>target_type 400<のプローブ/b>は、スキーマの統計を収集します。</li><li>target_type 500のプローブは、テーブルの統計を収集します。</li><li>次のプローブtarget_type 600は、インデックスの統計を収集します。</li><li>target_type 700のプローブは、シーケンスの統計を収集します。</li> <li>target_type 800のプローブは、ファンクションの統計を収集します。</li><li>target_type 900のプローブは、ファンクションの統計を収集します。view.</li></ul> |

**注釈**

この操作には認証が必要ありません

### get_*probe_config_server*{server_id}*database*{database_name}*schema*{schema_name}*view*{view_name}_{probe

コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/probe/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/view/{view_name}/{probe_id} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/probe/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/view/{view_name}/{probe_id}, headers␣
↪= headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch(
  "https://PEM-SERVER-IP/pem/api/v17/probe/config/server/{server_id}/database/
↪{database_name}/schema/{schema_name}/view/{view_name}/{probe_id}",
  {
    method: "GET",

    headers: headers,
  },
)
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

```
GET /probe/config/server/{server_id}/database/{database_name}/schema/{schema_name}/
view/{view_name}/{probe_id}
```

指定されたプローブ ID のビューレベルのプローブを取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| database_name | path | string | true | データベース名 |
| schema_name | path | string | true | スキーマ名 |
| view_name | path | string | true | ビュー名 |
| probe_id | path | integer | true | プローブ構成が返されるプローブ ID |
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例

```
{
  "probe_id": 0,
  "probe_name": "string",
  "default_interval_min": 0,
  "interval_min": 0,
  "default_interval_sec": 0,
  "interval_sec": 0,
  "default_interval": 0,
  "interval": 0,
  "default_lifetime": 0,
  "lifetime": 0,
  "default_enabled": true,
  "enabled": true,
  "target_type_id_returned": 0
}
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | プローブ |

**注釈**

この操作には認証が必要ありません

### Copy Probe

#### post_probe_copy_agent{agent_id}

コードサンプル

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v17/probe/copy/agent/{agent_id} \
  -H Content-Type: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Content-Type: application/json,
  X-Auth-Token: string
}

r = requests.post(https://PEM-SERVER-IP/pem/api/v17/probe/copy/agent/{agent_id},␣
↪headers = headers)

print(r.json())
```

```
const inputBody = [
  {
    "type": "string",
    "agent_id": 0
  }
];
const headers = {
  Content-Type:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v17/probe/copy/agent/{agent_id},
{
  method: POST,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

```
POST /probe/copy/agent/{agent_id}
```

エージェントレベルのプローブを別のエージェントにコピーします。

**ボディパラメーター**

```
[
  {
    "type": "string",
    "agent_id": 0
  }
]
```

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| agent_id | path | integer | true | エージェント ID |
| X-Auth-Token | header | string | true | 承認トークン |
| body | body | *CopyAgent-Data* | true | なし |

### 詳細な説明

**body** 注

- **type** 値は’agent’である必要があります

**応答例**

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ステータス 200 | None |

### レスポンススキーマ

**注釈**

この操作には認証が必要ありません

### post_probe_copy_server{server_id}

**コードサンプル**

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v17/probe/copy/server/{server_id} \
  -H Content-Type: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Content-Type: application/json,
  X-Auth-Token: string
}

r = requests.post(https://PEM-SERVER-IP/pem/api/v17/probe/copy/server/{server_id},␣
↪headers = headers)

print(r.json())
```

```
const inputBody = [
  {
    "type": "string",
    "server_id": 0
  }
];
const headers = {
  Content-Type:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v17/probe/copy/server/{server_id},
{
  method: POST,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

POST /probe/copy/server/{server_id}

サーバーレベルのプローブを別のサーバーにコピーします。

**ボディパラメーター**

```
[
  {
    "type": "string",
    "server_id": 0
  }
]
```

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| X-Auth-Token | header | string | true | 承認トークン |
| body | body | *CopyServer-Data* | true | なし |

### 詳細な説明

**body** 注

- **type** 値は’server’である必要があります

**応答例**

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ステータス 200 | None |

### レスポンススキーマ

**注釈**

この操作には認証が必要ありません

## post_probe_copy_server{server_id}database{database_name}

**コードサンプル**

```
#  You can also use wget
```

(次のページに続く)

(前のページからの続き)

```
curl -X POST https://PEM-SERVER-IP/pem/api/v17/probe/copy/server/{server_id}/database/
↪{database_name} \
  -H Content-Type: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Content-Type: application/json,
  X-Auth-Token: string
}

r = requests.post(https://PEM-SERVER-IP/pem/api/v17/probe/copy/server/{server_id}/
↪database/{database_name}, headers = headers)

print(r.json())
```

```
const inputBody = [
  {
    "type": "string",
    "server_id": 0,
    "database_name": "string"
  }
];
const headers = {
  Content-Type:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v17/probe/copy/server/{server_id}/database/
↪{database_name},
{
  method: POST,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

POST /probe/copy/server/{server_id}/database/{database_name}

データベースレベルのプローブを別のデータベースにコピーします。

**ボディパラメーター**

```
[
  {
    "type": "string",
    "server_id": 0,
    "database_name": "string"
  }
]
```

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| database_name | path | string | true | データベース名 |
| X-Auth-Token | header | string | true | 承認トークン |
| body | body | *Copy-Database-Data* | true | なし |

### 詳細な説明

**body** 注

- **type** 値は’database’である必要があります

**応答例**

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ステータス 200 | None |

### レスポンススキーマ

**注釈**

この操作には認証が必要ありません

## post_*probe_copy_server*{server_id}*database*{database_name}*schema*{schema_name}

**コードサンプル**

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v17/probe/copy/server/{server_id}/database/
↪{database_name}/schema/{schema_name} \
  -H Content-Type: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Content-Type: application/json,
  X-Auth-Token: string
}

r = requests.post(https://PEM-SERVER-IP/pem/api/v17/probe/copy/server/{server_id}/
↪database/{database_name}/schema/{schema_name}, headers = headers)

print(r.json())
```

```
const inputBody = [
  {
    "type": "string",
    "server_id": 0,
    "database_name": "string",
    "schema_name": "string"
  }
];
const headers = {
  Content-Type:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v17/probe/copy/server/{server_id}/database/
↪{database_name}/schema/{schema_name},
{
  method: POST,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

```
POST /probe/copy/server/{server_id}/database/{database_name}/schema/{schema_name}
```

スキーマレベルのプローブを別のスキーマにコピーします。

**ボディパラメーター**

```
[
  {
    "type": "string",
    "server_id": 0,
    "database_name": "string",
    "schema_name": "string"
  }
]
```

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| database_name | path | string | true | データベース名 |
| schema_name | path | string | true | スキーマ名 |
| X-Auth-Token | header | string | true | 承認トークン |
| body | body | *CopySchemaData* | true | なし |

### 詳細な説明

**body** 注

- **type** 値は’schema’である必要があります

**応答例**

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ステータス 200 | None |

### レスポンススキーマ

**注釈**

この操作には認証が必要ありません

### Probe data

#### get_*probe_data_agent*{agent_id}_{probe_id}

コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/probe/data/agent/{agent_id}/{probe_id} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/probe/data/agent/{agent_id}/{probe_
↪id}, headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch(
  "https://PEM-SERVER-IP/pem/api/v17/probe/data/agent/{agent_id}/{probe_id}",
  {
    method: "GET",

    headers: headers,
  },
)
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /probe/data/agent/{agent_id}/{probe_id}

エージェントの Probe data を取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| agent_id | path | integer | true | エージェント ID |
| probe_id | path | integer | true | データが返されるプローブ ID |
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例

```
[{}]
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | Inline |

### レスポンススキーマ

> **注釈**
>
> この操作には認証が必要ありません

## get_*probe_data_server*{server_id}_{probe_id}

### コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/probe/data/server/{server_id}/{probe_id}
↪ \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/probe/data/server/{server_id}/
```

(次のページに続く)

(前のページからの続き)

```
↪{probe_id}, headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch(
  "https://PEM-SERVER-IP/pem/api/v17/probe/data/server/{server_id}/{probe_id}",
  {
    method: "GET",

    headers: headers,
  },
)
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /probe/data/server/{server_id}/{probe_id}

サーバーの Probe data を取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| probe_id | path | integer | true | データが返されるプローブ ID |
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例

```
[{}]
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | Inline |

レスポンススキーマ

**注釈**

この操作には認証が必要ありません

### get_*probe_data_server*{server_id}*database*{database_name}_{probe_id}

コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/probe/data/server/{server_id}/database/
↪{database_name}/{probe_id} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/probe/data/server/{server_id}/
↪database/{database_name}/{probe_id}, headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch(
  "https://PEM-SERVER-IP/pem/api/v17/probe/data/server/{server_id}/database/{database_
↪name}/{probe_id}",
  {
    method: "GET",
```

(次のページに続く)

(前のページからの続き)

```
    headers: headers,
  },
)
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /probe/data/server/{server_id}/database/{database_name}/{probe_id}

データベースの Probe data を取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| database_name | path | string | true | データベース名 |
| probe_id | path | integer | true | データが返されるプローブ ID |
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例

```
[{}]
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | Inline |

### レスポンススキーマ

**注釈**

この操作には認証が必要ありません

### get_*probe_data_server*{server_id}*database*{database_name}*schema*{schema_name}_{probe_id}

コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/probe/data/server/{server_id}/database/
↪{database_name}/schema/{schema_name}/{probe_id} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/probe/data/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/{probe_id}, headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch(
  "https://PEM-SERVER-IP/pem/api/v17/probe/data/server/{server_id}/database/{database_
↪name}/schema/{schema_name}/{probe_id}",
  {
    method: "GET",

    headers: headers,
  },
)
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /probe/data/server/{server_id}/database/{database_name}/schema/{schema_name}/{probe_id}

スキーマの Probe data を取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| database_name | path | string | true | データベース名 |
| schema_name | path | string | true | スキーマ名 |
| probe_id | path | integer | true | データが返されるプローブ ID |
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例

```
[{}]
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | Inline |

### レスポンススキーマ

**注釈**

この操作には認証が必要ありません

## get_*probe_data_server*{server_id}*database*{database_name}*schema*{schema_name}*table*{table_name}_{probe_

### コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/probe/data/server/{server_id}/database/
↪{database_name}/schema/{schema_name}/table/{table_name}/{probe_id} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}
```

(次のページに続く)

(前のページからの続き)

```
r = requests.get(https://PEM-SERVER-IP/pem/api/v17/probe/data/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/table/{table_name}/{probe_id},␣
↪headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch(
  "https://PEM-SERVER-IP/pem/api/v17/probe/data/server/{server_id}/database/{database_
↪name}/schema/{schema_name}/table/{table_name}/{probe_id}",
  {
    method: "GET",

    headers: headers,
  },
)
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /probe/data/server/{server_id}/database/{database_name}/schema/{schema_name}/table/{table_name}/{probe_id}

テーブルの Probe data を取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| database_name | path | string | true | データベース名 |
| schema_name | path | string | true | スキーマ名 |
| table_name | path | string | true | テーブル名 |
| probe_id | path | integer | true | データが返されるプローブ ID |
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例

```
[{}]
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | Inline |

### レスポンススキーマ

> **注釈**
>
> この操作には認証が必要ありません

## get_*probe_data_server*{server_id}*database*{database_name}*schema*{schema_name}*index*{index_name}_{probe

### コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/probe/data/server/{server_id}/database/
↪{database_name}/schema/{schema_name}/index/{index_name}/{probe_id} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/probe/data/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/index/{index_name}/{probe_id},␣
↪headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};
```

(次のページに続く)

(前のページからの続き)

```
fetch(
  "https://PEM-SERVER-IP/pem/api/v17/probe/data/server/{server_id}/database/{database_
↪name}/schema/{schema_name}/index/{index_name}/{probe_id}",
  {
    method: "GET",

    headers: headers,
  },
)
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /probe/data/server/{server_id}/database/{database_name}/schema/{schema_name}/index/{index_name}/{probe_id}

インデックスの Probe data を取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| database_name | path | string | true | データベース名 |
| schema_name | path | string | true | スキーマ名 |
| index_name | path | string | true | インデックス名 |
| probe_id | path | integer | true | データが返されるプローブ ID |
| X-Auth-Token | header | string | true | 承認トークン |

#### 応答例

```
[{}]
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | Inline |

レスポンススキーマ

> **注釈**
>
> この操作には認証が必要ありません

**get_*probe_data_server*{server_id}*database*{database_name}*schema*{schema_name}*sequence*{sequence_name}**

コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/probe/data/server/{server_id}/database/
↪{database_name}/schema/{schema_name}/sequence/{sequence_name}/{probe_id} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/probe/data/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/sequence/{sequence_name}/{probe_id},␣
↪headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch(
  "https://PEM-SERVER-IP/pem/api/v17/probe/data/server/{server_id}/database/{database_
↪name}/schema/{schema_name}/sequence/{sequence_name}/{probe_id}",
  {
```

(次のページに続く)

(前のページからの続き)

```
    method: "GET",

    headers: headers,
  },
)
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /probe/data/server/{server_id}/database/{database_name}/schema/{schema_name}/sequence/{sequence_name}/{probe_id}

シーケンスの Probe data を取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| database_name | path | string | true | データベース名 |
| schema_name | path | string | true | スキーマ名 |
| sequence_name | path | string | true | シークエンス名 |
| probe_id | path | integer | true | データが返されるプローブ ID |
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例

```
[{}]
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | Inline |

### レスポンススキーマ

> **注釈**
>
> この操作には認証が必要ありません

#### get_*probe_data_server*{server_id}*database*{database_name}*schema*{schema_name}*view*{view_name}_{probe_i

#### コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/probe/data/server/{server_id}/database/
↪{database_name}/schema/{schema_name}/view/{view_name}/{probe_id} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/probe/data/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/view/{view_name}/{probe_id}, headers␣
↪= headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch(
  "https://PEM-SERVER-IP/pem/api/v17/probe/data/server/{server_id}/database/{database_
↪name}/schema/{schema_name}/view/{view_name}/{probe_id}",
  {
    method: "GET",

    headers: headers,
  },
)
```

(次のページに続く)

(前のページからの続き)

```
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /probe/data/server/{server_id}/database/{database_name}/schema/{schema_name}/view/{view_name}/{probe_id}

ビューの Probe data を取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| database_name | path | string | true | データベース名 |
| schema_name | path | string | true | スキーマ名 |
| view_name | path | string | true | ビュー名 |
| probe_id | path | integer | true | データが返されるプローブ ID |
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例

```
[{}]
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | Inline |

### レスポンススキーマ

**注釈**

この操作には認証が必要ありません

**get_*probe_data_server*{server_id}*database*{database_name}*schema*{schema_name}*function*{function_name}_{**

### コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/probe/data/server/{server_id}/database/
↪{database_name}/schema/{schema_name}/function/{function_name}/{probe_id} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/probe/data/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/function/{function_name}/{probe_id},␣
↪headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch(
  "https://PEM-SERVER-IP/pem/api/v17/probe/data/server/{server_id}/database/{database_
↪name}/schema/{schema_name}/function/{function_name}/{probe_id}",
  {
    method: "GET",

    headers: headers,
  },
)
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /probe/data/server/{server_id}/database/{database_name}/schema/{schema_name}/function/{function_name}/{probe_id}

ファンクションの Probe data を取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| database_name | path | string | true | データベース名 |
| schema_name | path | string | true | スキーマ名 |
| function_name | path | string | true | ファンクション名 |
| probe_id | path | integer | true | データが返されるプローブ ID |
| X-Auth-Token | header | string | true | 承認トークン |

#### 応答例

```
[{}]
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | Inline |

### レスポンススキーマ

> **注釈**
>
> この操作には認証が必要ありません

## get_*probe*{probe_id}*data*

#### コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/probe/{probe_id}/data/ \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}
```

(次のページに続く)

(前のページからの続き)

```
r = requests.get(https://PEM-SERVER-IP/pem/api/v17/probe/{probe_id}/data/, headers =␣
↪headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch("https://PEM-SERVER-IP/pem/api/v17/probe/{probe_id}/data/", {
  method: "GET",

  headers: headers,
})
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /probe/{probe_id}/data/

すべてのアクセス可能なサーバー/エージェントにわたって、指定されたプローブ ID の pemdata スキーマから現在のプローブデータを結果ます。

RBAC server_id は、pem.server RLS チームの可視性に対して検証されます。エージェントレベルのプローブの場合、server_id パラメーターは適用されません。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| probe_id | path | integer | true | 数値プローブ ID |
| server_id | query | integer | false | サーバー ID でフィルターします サーバーレベルのプローブのみ。 RBAC を適用します。 |
| offset | query | integer | false | スキップする行。オフセットがない場合は省略します。 |
| pageSize | query | integer | false | 返される最大行数 1〜1000。 all を返すのを省略します。 |
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例

```
[{}]
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | Inline |
| 400 | Bad Request | 不正な要求 — 無効なパラメーター | None |
| 404 | Not Found | プローブが見つからない、またはサーバーにアクセスできません | None |
| 500 | Internal Server Error | 内部サーバーエラー | None |

### レスポンススキーマ

> **注釈**
>
> この操作には認証が必要ありません

## get_*probe*{internal_name}*data*

### コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/probe/{internal_name}/data/ \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/probe/{internal_name}/data/,␣
↪headers = headers)
```

(次のページに続く)

(前のページからの続き)

```
print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch("https://PEM-SERVER-IP/pem/api/v17/probe/{internal_name}/data/", {
  method: "GET",

  headers: headers,
})
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /probe/{internal_name}/data/

指定されたプローブ internal_name の pemdata スキーマから現在のプローブデータを結果ます。

/probe/{probe_id}/data/と同等ですが、internal_name たとえば server_info、cpu_usage などでアドレス指定されます。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| internal_name | path | string | true | プローブの内部名たとえば server_info、cpu_usage |
| server_id | query | integer | false | サーバー ID でフィルターします サーバーレベルのプローブのみ。RBAC を適用します。 |
| offset | query | integer | false | スキップする行。オフセットがない場合は省略します。 |
| pageSize | query | integer | false | 返される最大行数 1〜1000。 all を返すのを省略します。 |
| X-Auth-Token | header | string | true | 承認トークン |

#### 応答例

```
[{}]
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | Inline |
| 400 | Bad Request | 不正な要求 — 無効なパラメーター | None |
| 404 | Not Found | プローブが見つからない、またはサーバーにアクセスできません | None |
| 500 | Internal Server Error | 内部サーバーエラー | None |

### レスポンススキーマ

**注釈**

この操作には認証が必要ありません

#### Probe history

#### get_*probe_history_agent*{agent_id}*from*{start_time}*to*{end_time}_{probe_id}

#### コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/probe/history/agent/{agent_id}/from/
↪{start_time}/to/{end_time}/{probe_id} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/probe/history/agent/{agent_id}/
↪from/{start_time}/to/{end_time}/{probe_id}, headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch(
  "https://PEM-SERVER-IP/pem/api/v17/probe/history/agent/{agent_id}/from/{start_time}/
↪to/{end_time}/{probe_id}",
  {
    method: "GET",

    headers: headers,
  },
)
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /probe/history/agent/{agent_id}/from/{start_time}/to/{end_time}/{probe_id}

エージェントの Probe history を取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| agent_id | path | integer | true | エージェント ID |
| start_time | path | string | true | プローブ履歴を取得するための ISO-8601 形式の開始時間 |
| end_time | path | string | true | プローブ履歴を取得するための ISO-8601 形式の終了時間 |
| probe_id | path | integer | true | データが返されるプローブ ID |
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例

```
[{}]
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | Inline |

レスポンススキーマ

**注釈**

この操作には認証が必要ありません

### get_*probe_history_server*{server_id}*from*{start_time}*to*{end_time}_{probe_id}

コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/probe/history/server/{server_id}/from/
↪{start_time}/to/{end_time}/{probe_id} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/probe/history/server/{server_id}/
↪from/{start_time}/to/{end_time}/{probe_id}, headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch(
  "https://PEM-SERVER-IP/pem/api/v17/probe/history/server/{server_id}/from/{start_
↪time}/to/{end_time}/{probe_id}",
  {
    method: "GET",
```

(次のページに続く)

(前のページからの続き)

```
    headers: headers,
  },
)
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /probe/history/server/{server_id}/from/{start_time}/to/{end_time}/{probe_id}

サーバーの Probe history を取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| start_time | path | string | true | プローブ履歴を取得するための ISO-8601 形式の開始時間 |
| end_time | path | string | true | プローブ履歴を取得するための ISO-8601 形式の終了時間 |
| probe_id | path | integer | true | データが返されるプローブ ID |
| X-Auth-Token | header | string | true | 承認トークン |

#### 応答例

```
[{}]
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | Inline |

### レスポンススキーマ

**注釈**

この操作には認証が必要ありません

### get_*probe_history_server*{server_id}*database*{database_name}*from*{start_time}*to*{end_time}_{probe_id}

コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/probe/history/server/{server_id}/
↪database/{database_name}/from/{start_time}/to/{end_time}/{probe_id} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/probe/history/server/{server_id}/
↪database/{database_name}/from/{start_time}/to/{end_time}/{probe_id}, headers =␣
↪headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch(
  "https://PEM-SERVER-IP/pem/api/v17/probe/history/server/{server_id}/database/
↪{database_name}/from/{start_time}/to/{end_time}/{probe_id}",
  {
    method: "GET",

    headers: headers,
  },
)
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

`GET /probe/history/server/{server_id}/database/{database_name}/from/{start_time}/to/{end_time}/{probe_id}`

データベースの `Probe history` を取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| database_name | path | string | true | データベース名 |
| start_time | path | string | true | プローブ履歴を取得するための ISO-8601 形式の開始時間 |
| end_time | path | string | true | プローブ履歴を取得するための ISO-8601 形式の終了時間 |
| probe_id | path | integer | true | データが返されるプローブ ID |
| X-Auth-Token | header | string | true | 承認トークン |

#### 応答例

```
[{}]
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | Inline |

### レスポンススキーマ

**注釈**

この操作には認証が必要ありません

#### get_*probe_history_server*{server_id}*database*{database_name}*schema*{schema_name}*from*{start_time}*to*{end_t

#### コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/probe/history/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/from/{start_time}/to/{end_time}/
```

(次のページに続く)

(前のページからの続き)

```
↪{probe_id} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/probe/history/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/from/{start_time}/to/{end_time}/
↪{probe_id}, headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch(
  "https://PEM-SERVER-IP/pem/api/v17/probe/history/server/{server_id}/database/
↪{database_name}/schema/{schema_name}/from/{start_time}/to/{end_time}/{probe_id}",
  {
    method: "GET",

    headers: headers,
  },
)
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /probe/history/server/{server_id}/database/{database_name}/schema/{schema_name}/from/{start_time}/to/{end_time}/{probe_id}

スキーマの Probe history を取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| database_name | path | string | true | データベース名 |
| schema_name | path | string | true | スキーマ名 |
| start_time | path | string | true | プローブ履歴を取得するための ISO-8601 形式の開始時間 |
| end_time | path | string | true | プローブ履歴を取得するための ISO-8601 形式の終了時間 |
| probe_id | path | integer | true | データが返されるプローブ ID |
| X-Auth-Token | header | string | true | 承認トークン |

#### 応答例

```
[{}]
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | Inline |

### レスポンススキーマ

**注釈**

この操作には認証が必要ありません

## get_*probe_history_server*{server_id}*database*{database_name}*schema*{schema_name}*table*{table_name}*from*{s

#### コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/probe/history/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/table/{table_name}/from/{start_time}/
↪to/{end_time}/{probe_id} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/probe/history/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/table/{table_name}/from/{start_time}/
↪to/{end_time}/{probe_id}, headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch(
  "https://PEM-SERVER-IP/pem/api/v17/probe/history/server/{server_id}/database/
↪{database_name}/schema/{schema_name}/table/{table_name}/from/{start_time}/to/{end_
↪time}/{probe_id}",
  {
    method: "GET",

    headers: headers,
  },
)
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /probe/history/server/{server_id}/database/{database_name}/schema/{schema_name}/table/{table_name}/from/{start_time}/to/{end_time}/{probe_id}

テーブルの Probe history を取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| database_name | path | string | true | データベース名 |

次のページに続く

表 522 – 前のページからの続き

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| schema_name | path | string | true | スキーマ名 |
| table_name | path | string | true | テーブル名 |
| start_time | path | string | true | プローブ履歴を取得するための ISO-8601 形式の開始時間 |
| end_time | path | string | true | プローブ履歴を取得するための ISO-8601 形式の終了時間 |
| probe_id | path | integer | true | データが返されるプローブ ID |
| X-Auth-Token | header | string | true | 承認トークン |

**応答例**

```
[{}]
```

**応答**

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | Inline |

**レスポンススキーマ**

**注釈**

この操作には認証が必要ありません

### get_*probe_history_server*{server_id}*database*{database_name}*schema*{schema_name}*index*{index_name}*from*{

**コードサンプル**

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/probe/history/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/index/{index_name}/from/{start_time}/
↪to/{end_time}/{probe_id} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
```

(次のページに続く)

(前のページからの続き)

```
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/probe/history/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/index/{index_name}/from/{start_time}/
↪to/{end_time}/{probe_id}, headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch(
  "https://PEM-SERVER-IP/pem/api/v17/probe/history/server/{server_id}/database/
↪{database_name}/schema/{schema_name}/index/{index_name}/from/{start_time}/to/{end_
↪time}/{probe_id}",
  {
    method: "GET",

    headers: headers,
  },
)
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /probe/history/server/{server_id}/database/{database_name}/schema/{schema_name}/index/{index_name}/from/{start_time}/to/{end_time}/{probe_id}

インデックスの Probe history を取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| database_name | path | string | true | データベース名 |
| schema_name | path | string | true | スキーマ名 |
| index_name | path | string | true | インデックス名 |

次のページに続く

表 524 – 前のページからの続き

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| start_time | path | string | true | プローブ履歴を取得するための ISO-8601 形式の開始時間 |
| end_time | path | string | true | プローブ履歴を取得するための ISO-8601 形式の終了時間 |
| probe_id | path | integer | true | データが返されるプローブ ID |
| X-Auth-Token | header | string | true | 承認トークン |

#### 応答例

```
[{}]
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | Inline |

### レスポンススキーマ

**注釈**

この操作には認証が必要ありません

## get_*probe_history_server*{server_id}*database*{database_name}*schema*{schema_name}*sequence*{sequence_nar

#### コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/probe/history/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/sequence/{sequence_name}/from/{start_
↪time}/to/{end_time}/{probe_id} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
```

(次のページに続く)

(前のページからの続き)

```
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/probe/history/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/sequence/{sequence_name}/from/{start_
↪time}/to/{end_time}/{probe_id}, headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch(
  "https://PEM-SERVER-IP/pem/api/v17/probe/history/server/{server_id}/database/
↪{database_name}/schema/{schema_name}/sequence/{sequence_name}/from/{start_time}/to/
↪{end_time}/{probe_id}",
  {
    method: "GET",

    headers: headers,
  },
)
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /probe/history/server/{server_id}/database/{database_name}/schema/{schema_name}/sequence/{sequence_name}/from/{start_time}/to/{end_time}/{probe_id}

シーケンスの Probe history を取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| database_name | path | string | true | データベース名 |
| schema_name | path | string | true | スキーマ名 |
| sequence_name | path | string | true | シークエンス名 |
| start_time | path | string | true | プローブ履歴を取得するための ISO-8601 形式の開始時間 |

次のページに続く

表 526 – 前のページからの続き

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| end_time | path | string | true | プローブ履歴を取得するための ISO-8601 形式の終了時間 |
| probe_id | path | integer | true | データが返されるプローブ ID |
| X-Auth-Token | header | string | true | 承認トークン |

**応答例**

```
[{}]
```

**応答**

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | Inline |

**レスポンススキーマ**

> **注釈**
>
> この操作には認証が必要ありません

**get_*probe_history_server*{server_id}*database*{database_name}*schema*{schema_name}*view*{view_name}*from*{st**

**コードサンプル**

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/probe/history/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/view/{view_name}/from/{start_time}/to/
↪{end_time}/{probe_id} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}
```

(次のページに続く)

(前のページからの続き)

```
r = requests.get(https://PEM-SERVER-IP/pem/api/v17/probe/history/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/view/{view_name}/from/{start_time}/to/
↪{end_time}/{probe_id}, headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch(
  "https://PEM-SERVER-IP/pem/api/v17/probe/history/server/{server_id}/database/
↪{database_name}/schema/{schema_name}/view/{view_name}/from/{start_time}/to/{end_
↪time}/{probe_id}",
  {
    method: "GET",

    headers: headers,
  },
)
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /probe/history/server/{server_id}/database/{database_name}/schema/{schema_name}/view/{view_name}/from/{start_time}/to/{end_time}/{probe_id}

ビューの Probe history を取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| database_name | path | string | true | データベース名 |
| schema_name | path | string | true | スキーマ名 |
| view_name | path | string | true | ビュー名 |
| start_time | path | string | true | プローブ履歴を取得するための ISO-8601 形式の開始時間 |
| end_time | path | string | true | プローブ履歴を取得するための ISO-8601 形式の終了時間 |

次のページに続く

表 528 – 前のページからの続き

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| probe_id | path | integer | true | データが返されるプローブ ID |
| X-Auth-Token | header | string | true | 承認トークン |

#### 応答例

```
[{}]
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | Inline |

### レスポンススキーマ

> **注釈**
>
> この操作には認証が必要ありません

## get_*probe_history_server*{server_id}*database*{database_name}*schema*{schema_name}*function*{function_name}

### コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/probe/history/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/function/{function_name}/from/{start_
↪time}/to/{end_time}/{probe_id} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/probe/history/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/function/{function_name}/from/{start_
```

(次のページに続く)

(前のページからの続き)

```
↪time}/to/{end_time}/{probe_id}, headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch(
  "https://PEM-SERVER-IP/pem/api/v17/probe/history/server/{server_id}/database/
↪{database_name}/schema/{schema_name}/function/{function_name}/from/{start_time}/to/
↪{end_time}/{probe_id}",
  {
    method: "GET",

    headers: headers,
  },
)
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /probe/history/server/{server_id}/database/{database_name}/schema/{schema_name}/function/{function_name}/from/{start_time}/to/{end_time}/{probe_id}

ファンクションの Probe history を取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| server_id | path | integer | true | サーバー ID |
| database_name | path | string | true | データベース名 |
| schema_name | path | string | true | スキーマ名 |
| function_name | path | string | true | ファンクション名 |
| start_time | path | string | true | プローブ履歴を取得するための ISO-8601 形式の開始時間 |
| end_time | path | string | true | プローブ履歴を取得するための ISO-8601 形式の終了時間 |
| probe_id | path | integer | true | データが返されるプローブ ID |

次のページに続く

表 530 – 前のページからの続き

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例

```
[{}]
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | Inline |

### レスポンススキーマ

**注釈**

この操作には認証が必要ありません

## get_*probe*{probe_id}*history_from*{start_time}*to*{end_time}

### コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/probe/{probe_id}/history/from/{start_
↪time}/to/{end_time} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/probe/{probe_id}/history/from/
↪{start_time}/to/{end_time}, headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch(
  "https://PEM-SERVER-IP/pem/api/v17/probe/{probe_id}/history/from/{start_time}/to/
↪{end_time}",
  {
    method: "GET",

    headers: headers,
  },
)
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /probe/{probe_id}/history/from/{start_time}/to/{end_time}

指定された時間範囲内の指定されたプローブ ID の pemhistory スキーマから履歴プローブデータを結果ます。

resume_history = true の場合、空のリストを返します。 RBAC データエンドポイントと同じ。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| probe_id | path | integer | true | 数値プローブ ID |
| start_time | path | string | true | Recorded_time の下限両端を含みます。 ISO 8601 タイムスタンプ *YYYY-MM-DD<sep>HH:MM:SS[.ffffff][TZ]*、*<sep>* **は** *`T`* **またはスペースであり、オプションの** *`TZ`* **は** `Z`、*±HH*、または `±HH:MM`です。例 *2024-01-01T00:00:00Z*、*2024-01-01 00:00:00+00* PostgreSQL のデフォルト。 |

次のページに続く

表 532 – 前のページからの続き

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| end_time | path | string | true | Recorded_time の上限値を含みます。 ISO 8601 タイムスタンプ *YYYY-MM-DD<sep>HH:MM:SS[.ffffff][TZ]*、*<sep>* は `T` またはスペースであり、オプションの `TZ` は `Z`、*±HH*、または `±HH:MM` です。例 *2024-12-31T23:59:59Z*、*2024-12-31 23:59:59+00* PostgreSQL のデフォルト。 |
| server_id | query | integer | false | サーバー ID でフィルターします サーバーレベルのプローブのみ。RBAC を適用します。 |
| offset | query | integer | false | スキップする行。オフセットがない場合は省略します。 |
| pageSize | query | integer | false | 返される最大行数 1〜1000。 all を返すのを省略します。 |
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例

```
[{}]
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | Inline |
| 400 | Bad Request | 不正な要求 — 無効なパラメーター | None |
| 404 | Not Found | プローブが見つからない、またはサーバーにアクセスできません | None |
| 500 | Internal Server Error | 内部サーバーエラー | None |

レスポンススキーマ

> **注釈**
>
> この操作には認証が必要ありません

### get_*probe*{internal_name}*history_from*{start_time}*to*{end_time}

コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/probe/{internal_name}/history/from/
↪{start_time}/to/{end_time} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/probe/{internal_name}/history/from/
↪{start_time}/to/{end_time}, headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch(
  "https://PEM-SERVER-IP/pem/api/v17/probe/{internal_name}/history/from/{start_time}/
↪to/{end_time}",
  {
    method: "GET",

    headers: headers,
  },
)
  .then(function (res) {
    return res.json();
  })
```

(次のページに続く)

(前のページからの続き)

```
  .then(function (body) {
    console.log(body);
  });
```

GET /probe/{internal_name}/history/from/{start_time}/to/{end_time}

指定された時間範囲内の指定されたプローブ internal_name の pemhistory スキーマからプローブ履歴データを結果ます。

/probe/{probe_id}/history/…と同等ですが、internal_name でアドレス指定されます。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| internal_name | path | string | true | プローブの内部名たとえば server_info、cpu_usage |
| start_time | path | string | true | Recorded_time の下限両端を含みます。 ISO 8601 タイムスタンプ *YYYY-MM-DD<sep>HH:MM:SS[.ffffff][TZ]*、**<sep> は `T` またはスペースであり、オプションの `TZ` は** `Z`、*±HH*、または `±HH:MM` です。例 *2024-01-01T00:00:00Z*、*2024-01-01 00:00:00+00* PostgreSQL のデフォルト。 |
| end_time | path | string | true | Recorded_time の上限値を含みます。 ISO 8601 タイムスタンプ *YYYY-MM-DD<sep>HH:MM:SS[.ffffff][TZ]*、**<sep> は `T` またはスペースであり、オプションの `TZ` は** `Z`、*±HH*、または `±HH:MM` です。例 *2024-12-31T23:59:59Z*、*2024-12-31 23:59:59+00* PostgreSQL のデフォルト。 |
| server_id | query | integer | false | サーバー ID でフィルターします サーバーレベルのプローブのみ。RBAC を適用します。 |
| offset | query | integer | false | スキップする行。オフセットがない場合は省略します。 |
| pageSize | query | integer | false | 返される最大行数 1〜1000。 all を返すのを省略します。 |
| X-Auth-Token | header | string | true | 承認トークン |

#### 応答例

```
[{}]
```

#### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 成功した応答 | Inline |
| 400 | Bad Request | 不正な要求 — 無効なパラメーター | None |
| 404 | Not Found | プローブが見つからない、またはサーバーにアクセスできません | None |
| 500 | Internal Server Error | 内部サーバーエラー | None |

#### レスポンススキーマ

**注釈**

この操作には認証が必要ありません

### Webhook

#### get_*webhook*

#### コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/webhook/ \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
```

(次のページに続く)

(前のページからの続き)

```
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/webhook/, headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch("https://PEM-SERVER-IP/pem/api/v17/webhook/", {
  method: "GET",

  headers: headers,
})
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /webhook/

*Webhook* リストを取得します。

Webhook のリストを返します。 name 完全一致または name_contains 部分文字列検索クエリパラメーターを介してオプションの名前ベースのフィルタリングをサポートします。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| X-Auth-Token | header | string | true | 承認トークン |
| name | query | string | false | 完全な名前の一致大文字と小文字を区別して Webhook をフィルタリングします。 name_contains と相互排他的。 |
| name_contains | query | string | false | 部分文字列の一致大文字と小文字を区別して Webhook をフィルタリングします。 name と相互排他的。 |

### 応答例

```
[
  {
    "id": 0,
    "name": "string",
    "url": "string",
    "enabled": true,
    "method": "POST",
    "http_headers": [
      {
        "http_header_id": 0,
        "http_header_key": "string",
        "http_header_value": "string"
      }
    ],
    "payload_template": "{text: You have new alert from PEM, attachments: [{text:␣
↪AlertID \n%AlertID% \nAlertName \n%AlertName%}]}",
    "low_alert": true,
    "med_alert": true,
    "high_alert": true,
    "cleared_alert": true,
    "payload_type": "ALERT"
  }
]
```


応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ステータス 200 | Inline |

次のページに続く

表 537 – 前のページからの続き

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 400 | Bad Request | 相互排他性違反 `name`または類似の完全一致フィルタと`name_contains`または類似の部分文字列フィルタが両方とも同じ要求で指定されました。フィルタリングは、ペアのうちの 1 つのみを受け入れます。 | Inline |

### レスポンススキーマ

ステータスコード **200**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| _anonymous_ | *Platform Compatibility* ] | false | none | なし |
| » id | integer | false | none | Webhook ID |
| » name | string | false | none | Webhook 名 |
| » url | string | false | none | Webhook URL |
| » enabled | boolean | false | none | Webhook が有効になっているかどうかを判断します。 |
| » method | string | false | none | リクエストメソッド |
| » http_headers | *Platform Compatibility* ] | false | none | Webhook で使用される HTTP ヘッダーのキー/値ペアのリスト |
| » http_header_id | integer | false | none | HTTP ヘッダーの ID |
| » http_header_key | string | false | none | HTTP ヘッダーキーの名前 |
| » http_header_value | string | false | none | HTTP ヘッダーキーの名前 |
| » payload_template | string | false | none | ペイロードテンプレート。以下は、Slack のペイロードの例です。 |
| » low_alert | boolean | false | none | 低アラートを有効にするかどうかを決定します。 |
| » med_alert | boolean | false | none | 中程度のアラートを有効にするかどうかを決定します。 |
| » high_alert | boolean | false | none | 厳重なアラートを有効にするかどうかを決定します。 |

次のページに続く

表 538 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » cleared_alert | boolean | false | none | クリアされたアラートが有効かどうかを判断します。 |
| » payload_type | string | false | none | ペイロードタイプ |

### 列挙値

| Property | Value |
|---|---|
| method | POST |
| method | PUT |
| payload_type | ALERT |
| payload_type | JOB |

ステータスコード **400**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » success | integer | false | none | なし |
| » errormsg | string | false | none | なし |

**注釈**

この操作には認証が必要ありません

### post_*webhook*

#### コードサンプル

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v17/webhook/ \
  -H Content-Type: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Content-Type: application/json,
  X-Auth-Token: string
}

r = requests.post(https://PEM-SERVER-IP/pem/api/v17/webhook/, headers = headers)
```

(次のページに続く)

(前のページからの続き)

```
print(r.json())
```

```
const inputBody = {
  "name": "string",
  "url": "string",
  "enabled": true,
  "method": "POST",
  "http_headers": [
    {
      "http_header_key": "string",
      "http_header_value": "string"
    }
  ],
  "payload_template": "{ Alert Name:%AlertName%,type:Webhook }",
  "low_alert": true,
  "med_alert": true,
  "high_alert": true,
  "cleared_alert": true,
  "payload_type": "ALERT"
};
const headers = {
  Content-Type:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v17/webhook/,
{
  method: POST,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

POST /webhook/

新しい *Webhook* を追加します。

ボディパラメーター

```
{
  "name": "string",
```

(次のページに続く)

(前のページからの続き)


```
  "url": "string",
  "enabled": true,
  "method": "POST",
  "http_headers": [
    {
      "http_header_key": "string",
      "http_header_value": "string"
    }
  ],
  "payload_template": "{ Alert Name:%AlertName%,type:Webhook }",
  "low_alert": true,
  "med_alert": true,
  "high_alert": true,
  "cleared_alert": true,
  "payload_type": "ALERT"
}
```


### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| X-Auth-Token | header | string | true | 承認トークン |
| body | body | *Webhook-PostData* | true | なし |

### 応答例

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | Webhook が正常に作成されました。 | None |

### レスポンススキーマ

**注釈**

この操作には認証が必要ありません

### get_*webhook*{webhook_id}

コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/webhook/{webhook_id} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/webhook/{webhook_id}, headers =␣
↪headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch("https://PEM-SERVER-IP/pem/api/v17/webhook/{webhook_id}", {
  method: "GET",

  headers: headers,
})
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /webhook/{webhook_id}

*webhook_id* で識別される *Webhook* の情報を取得します。

パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| webhook_id | path | string | true | なし |
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例

```
{
  "id": 0,
  "name": "string",
  "url": "string",
  "enabled": true,
  "method": "POST",
  "http_headers": [
    {
      "http_header_id": 0,
      "http_header_key": "string",
      "http_header_value": "string"
    }
  ],
  "payload_template": "{text: You have new alert from PEM, attachments: [{text:␣
↪AlertID \n%AlertID% \nAlertName \n%AlertName%}]}",
  "low_alert": true,
  "med_alert": true,
  "high_alert": true,
  "cleared_alert": true,
  "payload_type": "ALERT"
}
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ステータス 200 | *Webhooks* |

**注釈**

この操作には認証が必要ありません

### put_*webhook*{webhook_id}

コードサンプル

```
#  You can also use wget

curl -X PUT https://PEM-SERVER-IP/pem/api/v17/webhook/{webhook_id} \
  -H Content-Type: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Content-Type: application/json,
  X-Auth-Token: string
}

r = requests.put(https://PEM-SERVER-IP/pem/api/v17/webhook/{webhook_id}, headers =␣
↪headers)

print(r.json())
```

```
const inputBody = {
  "name": "string",
  "url": "string",
  "enabled": true,
  "method": "POST",
  "http_headers": {
    "added": [
      {
        "http_header_key": "string",
        "http_header_value": "string"
      }
    ],
    "changed": [
      {
        "http_header_id": 0,
        "http_header_key": "string",
        "http_header_value": "string"
      }
    ],
    "deleted": [
      {
        "http_header_id": 0
      }
    ],
    "payload_type": "ALERT"
```

(次のページに続く)

(前のページからの続き)

```
  },
  "payload_template": "{text: You have new alert from PEM, attachments: [{text:␣
↪AlertID \n%AlertID% \nAlertName \n%AlertName%}]}",
  "low_alert": true,
  "med_alert": true,
  "high_alert": true,
  "cleared_alert": true
};
const headers = {
  Content-Type:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v17/webhook/{webhook_id},
{
  method: PUT,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

PUT /webhook/{webhook_id}

*Webhook* 情報を更新します。

ボディパラメーター

```
{
  "name": "string",
  "url": "string",
  "enabled": true,
  "method": "POST",
  "http_headers": {
    "added": [
      {
        "http_header_key": "string",
        "http_header_value": "string"
      }
    ],
    "changed": [
      {
        "http_header_id": 0,
```

(次のページに続く)

(前のページからの続き)


```
        "http_header_key": "string",
        "http_header_value": "string"
      }
    ],
    "deleted": [
      {
        "http_header_id": 0
      }
    ],
    "payload_type": "ALERT"
  },
  "payload_template": "{text: You have new alert from PEM, attachments: [{text:␣
↪AlertID \n%AlertID% \nAlertName \n%AlertName%}]}",
  "low_alert": true,
  "med_alert": true,
  "high_alert": true,
  "cleared_alert": true
}
```


### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| webhook_id | path | string | true | なし |
| X-Auth-Token | header | string | true | 承認トークン |
| body | body | *WebhookPutData* | true | なし |

### 応答例

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | Webhook が正常に更新されました。 | None |

レスポンススキーマ

> **注釈**
>
> この操作には認証が必要ありません

### delete_*webhook*{webhook_id}

コードサンプル

```
#  You can also use wget

curl -X DELETE https://PEM-SERVER-IP/pem/api/v17/webhook/{webhook_id} \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  X-Auth-Token: string
}

r = requests.delete(https://PEM-SERVER-IP/pem/api/v17/webhook/{webhook_id}, headers =␣
↪headers)

print(r.json())
```

```
const headers = {
  "X-Auth-Token": "string",
};

fetch("https://PEM-SERVER-IP/pem/api/v17/webhook/{webhook_id}", {
  method: "DELETE",

  headers: headers,
})
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

```
DELETE /webhook/{webhook_id}
```

*Webhook* を削除します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| webhook_id | path | string | true | なし |
| X-Auth-Token | header | string | true | 承認トークン |

#### 応答例

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | Webhook は正常に削除されました | None |

### レスポンススキーマ

> **注釈**
>
> この操作には認証が必要ありません

## Email Groups

### get__email_groups

#### コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/email/groups \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/email/groups, headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch("https://PEM-SERVER-IP/pem/api/v17/email/groups", {
  method: "GET",

  headers: headers,
})
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /email/groups

**メールグループリストを取得します。**

電子メールグループのリストを返します。 name 完全一致または name_contains 部分文字列検索クエリパラメーターを介してオプションの名前ベースのフィルタリングをサポートします。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| X-Auth-Token | header | string | true | 承認トークン |
| name | query | string | false | 名前の完全な一致大文字と小文字を区別してメールグループをフィルタリングします。 name_contains と相互排他的。 |
| name_contains | query | string | false | 部分文字列の一致大文字と小文字を区別してメールグループをフィルタリングします。 name と相互排他的。 |

**応答例**

```
[
  {
    "id": 0,
    "oid": 0,
    "gid": 0,
    "name": "string",
```

(次のページに続く)

(前のページからの続き)


```
    "to_addr": "string",
    "cc_addr": "string",
    "bcc_addr": "string",
    "from_addr": "string",
    "reply_to_addr": "string",
    "subject_prefix": "string",
    "from_time": "00:00:00",
    "to_time": "23:59:59"
  }
]
```


### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ステータス 200 | Inline |
| 400 | Bad Request | 相互排他性違反 `name`または類似の完全一致フィルタと`name_contains`または類似の部分文字列フィルタが両方とも同じ要求で指定されました。フィルタリングは、ペアのうちの 1 つのみを受け入れます。 | Inline |

### レスポンススキーマ

ステータスコード **200**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| _anonymous_ | *Platform Compatibility* ] | false | none | なし |
| » id | integer | false | none | メールグループ ID |
| » oid | integer | false | none | メールグループオプション ID |

次のページに続く

表 551 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » gid | integer | false | none | メールグループ ID |
| » name | string | false | none | メールグループ名 |
| » to_addr | string | false | none | メールグループオプションの受信者アドレス |
| » cc_addr | string | false | none | メールグループオプション CC アドレス |
| » bcc_addr | string | false | none | メールグループオプション BCC アドレス |
| » from_addr | string | false | none | メールグループオプションの送信者アドレス |
| » reply_to_addr | string | false | none | メールグループオプションの返信先アドレス |
| » subject_prefix | string | false | none | メールグループオプションの件名プレフィックス |
| » from_time | string | false | none | メンバーの時間から選択した時間範囲までのメールグループオプション |
| » to_time | string | false | none | メンバーの時間範囲を選択するメールグループオプション |

ステータスコード **400**

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » success | integer | false | none | なし |
| » errormsg | string | false | none | なし |

**注釈**

この操作には認証が必要ありません

### post__email_groups

#### コードサンプル

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v17/email/groups \
  -H Content-Type: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Content-Type: application/json,
  X-Auth-Token: string
}

r = requests.post(https://PEM-SERVER-IP/pem/api/v17/email/groups, headers = headers)

print(r.json())
```

```
const inputBody = {
  "name": "string",
  "options": [
    {
      "to_addr": "string",
      "cc_addr": "string",
      "bcc_addr": "string",
      "from_addr": "string",
      "reply_to_addr": "string",
      "subject_prefix": "string",
      "from_time": "00:00:00",
      "to_time": "23:59:59"
    }
  ]
};
const headers = {
  Content-Type:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v17/email/groups,
{
  method: POST,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

```
POST /email/groups
```

新しいメールグループを追加します。

ボディパラメーター

```
{
  "name": "string",
  "options": [
    {
      "to_addr": "string",
      "cc_addr": "string",
      "bcc_addr": "string",
      "from_addr": "string",
      "reply_to_addr": "string",
      "subject_prefix": "string",
      "from_time": "00:00:00",
      "to_time": "23:59:59"
    }
  ]
}
```


#### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| X-Auth-Token | header | string | true | 承認トークン |
| body | body | *Email-GroupsPost-Data* | true | なし |

#### 応答例

#### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 電子メールグループが正常に作成されました | None |

#### レスポンススキーマ

**注釈**

この操作には認証が必要ありません

### get_*email_groups*{email_group_id}

#### コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/email/groups/{email_group_id} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/email/groups/{email_group_id},␣
↪headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch("https://PEM-SERVER-IP/pem/api/v17/email/groups/{email_group_id}", {
  method: "GET",

  headers: headers,
})
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

GET /email/groups/{email_group_id}

*mail_group_id* で識別される、メールグループの情報を取得します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| email_group_id | path | string | true | なし |
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例

```
{
  "id": 0,
  "oid": 0,
  "gid": 0,
  "name": "string",
  "to_addr": "string",
  "cc_addr": "string",
  "bcc_addr": "string",
  "from_addr": "string",
  "reply_to_addr": "string",
  "subject_prefix": "string",
  "from_time": "00:00:00",
  "to_time": "23:59:59"
}
```

応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ステータス 200 | メールグループ |

**注釈**

この操作には認証が必要ありません

### put_*email_groups*{email_group_id}

コードサンプル

```
#  You can also use wget

curl -X PUT https://PEM-SERVER-IP/pem/api/v17/email/groups/{email_group_id} \
  -H Content-Type: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Content-Type: application/json,
  X-Auth-Token: string
}
```

(次のページに続く)

(前のページからの続き)

```
r = requests.put(https://PEM-SERVER-IP/pem/api/v17/email/groups/{email_group_id},␣
↪headers = headers)

print(r.json())
```

```
const inputBody = {
  "name": "string",
  "options": {
    "added": [
      {
        "to_addr": "string",
        "cc_addr": "string",
        "bcc_addr": "string",
        "from_addr": "string",
        "reply_to_addr": "string",
        "subject_prefix": "string",
        "from_time": "00:00:00",
        "to_time": "23:59:59"
      }
    ],
    "changed": [
      {
        "oid": 0,
        "to_addr": "string",
        "cc_addr": "string",
        "bcc_addr": "string",
        "from_addr": "string",
        "reply_to_addr": "string",
        "subject_prefix": "string",
        "from_time": "00:00:00",
        "to_time": "23:59:59"
      }
    ],
    "deleted": [
      {
        "oid": 0
      }
    ]
  }
};
const headers = {
  Content-Type:application/json,
  X-Auth-Token:string
};
```

(次のページに続く)

(前のページからの続き)

```
fetch(https://PEM-SERVER-IP/pem/api/v17/email/groups/{email_group_id},
{
  method: PUT,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

PUT /email/groups/{email_group_id}

メールグループ情報を更新します。

ボディパラメーター

```
{
  "name": "string",
  "options": {
    "added": [
      {
        "to_addr": "string",
        "cc_addr": "string",
        "bcc_addr": "string",
        "from_addr": "string",
        "reply_to_addr": "string",
        "subject_prefix": "string",
        "from_time": "00:00:00",
        "to_time": "23:59:59"
      }
    ],
    "changed": [
      {
        "oid": 0,
        "to_addr": "string",
        "cc_addr": "string",
        "bcc_addr": "string",
        "from_addr": "string",
        "reply_to_addr": "string",
        "subject_prefix": "string",
        "from_time": "00:00:00",
        "to_time": "23:59:59"
      }
```

(次のページに続く)

(前のページからの続き)


```
    ],
    "deleted": [
      {
        "oid": 0
      }
    ]
  }
}
```


### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| email_group_id | path | string | true | なし |
| X-Auth-Token | header | string | true | 承認トークン |
| body | body | *Email-GroupsPut-Data* | true | なし |

### 応答例

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 除外されたデータベースが正常に追加されました | None |

### レスポンススキーマ

**注釈**

この操作には認証が必要ありません

## delete_*email_groups*{email_group_id}

### コードサンプル

```
#  You can also use wget

curl -X DELETE https://PEM-SERVER-IP/pem/api/v17/email/groups/{email_group_id} \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  X-Auth-Token: string
}

r = requests.delete(https://PEM-SERVER-IP/pem/api/v17/email/groups/{email_group_id},␣
↪headers = headers)

print(r.json())
```

```
const headers = {
  "X-Auth-Token": "string",
};

fetch("https://PEM-SERVER-IP/pem/api/v17/email/groups/{email_group_id}", {
  method: "DELETE",

  headers: headers,
})
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

DELETE /email/groups/{email_group_id}

メールグループを削除します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| email_group_id | path | string | true | なし |
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 電子メールグループは正常に削除されました | None |

レスポンススキーマ

> **注釈**
>
> この操作には認証が必要ありません

>-

**get_*server_exclude_database*{sid}**

コードサンプル

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v17/server/exclude_database/{sid} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v17/server/exclude_database/{sid},␣
↪headers = headers)

print(r.json())
```

```
const headers = {
  Accept: "application/json",
  "X-Auth-Token": "string",
};

fetch("https://PEM-SERVER-IP/pem/api/v17/server/exclude_database/{sid}", {
  method: "GET",

  headers: headers,
})
```

(次のページに続く)

(前のページからの続き)

```
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

`GET /server/exclude_database/{sid}`

**サーバー *ID* で識別される、除外されたデータベースの情報を取得します。**

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| sid | path | string | true | サーバー ID |
| X-Auth-Token | header | string | true | 承認トークン |

**応答例**

```
{
  "server_id": 0,
  "name": "string",
  "exclude_databases": ["string"]
}
```

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | ステータス 200 | *ExcludeDatabases* |

**注釈**

この操作には認証が必要ありません

### put_*server_exclude_database*{sid}

**コードサンプル**

```
#  You can also use wget

curl -X PUT https://PEM-SERVER-IP/pem/api/v17/server/exclude_database/{sid} \
  -H Content-Type: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Content-Type: application/json,
  X-Auth-Token: string
}

r = requests.put(https://PEM-SERVER-IP/pem/api/v17/server/exclude_database/{sid},␣
↪headers = headers)

print(r.json())
```

```
const inputBody = {
  "exclude_databases": [
    "string"
  ]
};
const headers = {
  Content-Type:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v17/server/exclude_database/{sid},
{
  method: PUT,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

PUT /server/exclude_database/{sid}

除外されたデータベース情報を更新します。

ボディパラメーター

```
{
```

(次のページに続く)

(前のページからの続き)

```
  "exclude_databases": ["string"]
}
```

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| sid | path | string | true | サーバー ID |
| X-Auth-Token | header | string | true | 承認トークン |
| body | body | *Exclude-Databas-esPutData* | true | なし |

### 応答例

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 電子メールグループが正常に更新されました | None |

### レスポンススキーマ

**注釈**

この操作には認証が必要ありません

## delete_*server_exclude_database*{sid}

### コードサンプル

```
#  You can also use wget

curl -X DELETE https://PEM-SERVER-IP/pem/api/v17/server/exclude_database/{sid} \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  X-Auth-Token: string
}
```

(次のページに続く)

(前のページからの続き)

```
r = requests.delete(https://PEM-SERVER-IP/pem/api/v17/server/exclude_database/{sid},␣
↪headers = headers)

print(r.json())
```

```
const headers = {
  "X-Auth-Token": "string",
};

fetch("https://PEM-SERVER-IP/pem/api/v17/server/exclude_database/{sid}", {
  method: "DELETE",

  headers: headers,
})
  .then(function (res) {
    return res.json();
  })
  .then(function (body) {
    console.log(body);
  });
```

`DELETE /server/exclude_database/{sid}`

除外されたデータベースを削除します。

### パラメーター

| Name | In | Type | Required | Description |
|---|---|---|---|---|
| sid | path | string | true | サーバー ID |
| X-Auth-Token | header | string | true | 承認トークン |

### 応答例

### 応答

| Status | Meaning | Description | Schema |
|---|---|---|---|
| 200 | コマンドラインからのサーバーインストーラーの呼び出し | 除外されたデータベースは正常に削除されました | None |

レスポンススキーマ

> **注釈**
>
> この操作には認証が必要ありません

### Schemas

プロファイル


```
{
  "id": 0,
  "name": "string",
  "description": "string",
  "target_kind": "s",
  "published": true,
  "parent_id": 0,
  "draft_id": 0,
  "target_probe_configs": [
    {
      "probe_id": 0,
      "probe_name": "string",
      "enabled": true,
      "interval": 0,
      "lifetime": 0,
      "default_enabled": true,
      "default_interval": 0,
      "default_lifetime": 0,
      "target_type": 0,
      "force_enabled": true
    }
  ],
  "target_alert_configs": [
    {
      "id": 0,
      "alert_name": "string",
      "alert_template": 0,
      "description": "string",
      "enabled": true,
      "history_retention": 0,
      "frequency_min": 0,
      "auto_created": true,
      "template_id": "string",
      "params": [],
```


(次のページに続く)

(前のページからの続き)


```
      "param_names": ["string"],
      "param_types": "string",
      "params_units": ["string"],
      "threshold_unit": "string",
      "operator": "string",
      "low_threshold_value": 0,
      "medium_threshold_value": 0,
      "high_threshold_value": 0,
      "send_email": true,
      "all_alert_enable": true,
      "email_group_id": 0,
      "low_alert_enable": true,
      "low_email_group_id": 0,
      "med_alert_enable": true,
      "med_email_group_id": 0,
      "high_alert_enable": true,
      "high_email_group_id": 0,
      "send_trap": true,
      "snmp_trap_version": 0,
      "low_send_trap": true,
      "med_send_trap": true,
      "high_send_trap": true,
      "cleared_alert_enable": true,
      "execute_script": true,
      "execute_script_on_clear": true,
      "execute_script_on_pem_server": true,
      "script_code": "string",
      "submit_to_nagios": true,
      "override_default_config": true,
      "send_notification": true,
      "low_webhook_ids": [0],
      "med_webhook_ids": [0],
      "high_webhook_ids": [0],
      "cleared_webhook_ids": [0]
    }
  ]
}
```


プローブとアラート構成を含む完全なプロファイルの詳細 GET /profiles/{profile_id}によって返されます。

### プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| id | integer | false | none | なし |
| name | string | false | none | なし |
| description | string | false | none | なし |
| target_kind | string | false | none | プロファイルのターゲットタイプサーバーの場合は's'、エージェントの場合は'a'。 |
| published | boolean | false | none | プロファイルが公開される場合は true、ドラフトの場合は false。 |
| parent_id | integer | false | none | 親プロファイル ID 公開されたプロファイルの場合は null、下書きに設定されます。 |
| draft_id | integer | false | none | 下書きプロファイル ID 下書きがある公開済みプロファイルにのみ存在します。 |
| target_probe_configs | [ProbeConfig ] | false | none | プロファイルに関連付けられたプローブ構成。 |
| target_alert_configs | [Alert ] | false | none | プロファイルに関連付けられているアラート構成。 |

### 列挙値

| Property | Value |
|---|---|
| target_kind | s |
| target_kind | a |

### プロフィールドラフト

```
{
  "id": 0,
  "name": "string",
  "description": "string",
  "target_kind": "s",
  "published": true,
  "target_probe_configs": [
    {
      "probe_id": 0,
      "probe_name": "string",
      "enabled": true,
```

(次のページに続く)

(前のページからの続き)


```
      "interval": 0,
      "lifetime": 0,
      "default_enabled": true,
      "default_interval": 0,
      "default_lifetime": 0,
      "target_type": 0,
      "force_enabled": true
    }
  ],
  "target_alert_configs": [
    {
      "id": 0,
      "alert_name": "string",
      "alert_template": 0,
      "description": "string",
      "enabled": true,
      "history_retention": 0,
      "frequency_min": 0,
      "auto_created": true,
      "template_id": "string",
      "params": [],
      "param_names": ["string"],
      "param_types": "string",
      "params_units": ["string"],
      "threshold_unit": "string",
      "operator": "string",
      "low_threshold_value": 0,
      "medium_threshold_value": 0,
      "high_threshold_value": 0,
      "send_email": true,
      "all_alert_enable": true,
      "email_group_id": 0,
      "low_alert_enable": true,
      "low_email_group_id": 0,
      "med_alert_enable": true,
      "med_email_group_id": 0,
      "high_alert_enable": true,
      "high_email_group_id": 0,
      "send_trap": true,
      "snmp_trap_version": 0,
      "low_send_trap": true,
      "med_send_trap": true,
      "high_send_trap": true,
      "cleared_alert_enable": true,
      "execute_script": true,
```


(次のページに続く)

(前のページからの続き)


```
      "execute_script_on_clear": true,
      "execute_script_on_pem_server": true,
      "script_code": "string",
      "submit_to_nagios": true,
      "override_default_config": true,
      "send_notification": true,
      "low_webhook_ids": [0],
      "med_webhook_ids": [0],
      "high_webhook_ids": [0],
      "cleared_webhook_ids": [0]
    }
  ]
}
```


ドラフトプロファイルの詳細 GET /profiles/{profile_id}/draft によって返されます。parent_id または draft_id は含まれません。

### プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| id | integer | false | none | なし |
| name | string | false | none | なし |
| description | string | false | none | なし |
| target_kind | string | false | none | プロファイルのターゲットタイプサーバーの場合は's'、エージェントの場合は'a'。 |
| published | boolean | false | none | ドラフトの場合は常にfalse。 |
| target_probe_configs | [ProbeConfig ] | false | none | ドラフトに関連付けられたプローブ構成。 |
| target_alert_configs | [Alert ] | false | none | ドラフトに関連付けられたアラート構成。 |

### 列挙値

| Property | Value |
|---|---|
| target_kind | s |
| target_kind | a |

### プロフィール概要


```
{
  "id": 34,
  "name": "Production Agents",
  "description": "Monitoring production agents",
  "target_kind": "a",
  "active_assignments": 0,
  "published": true
}
```


リスト操作のトリミングされたプロファイル表現形式プローブ/アラート構成配列なし。

### プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| id | integer | false | none | なし |
| name | string | false | none | なし |
| description | string | false | none | なし |
| target_kind | string | false | none | プロファイルのターゲットタイプサーバーの場合は's'、エージェントの場合は'a'。 |
| published | boolean | false | none | コンビニエンスフラグステータスが'published'の場合に true。 |
| active_assignments | integer | false | none | このプロファイルに割り当てられたアクティブなサーバー/エージェントの数。 |

### 列挙値

| Property | Value |
|---|---|
| target_kind | s |
| target_kind | a |

### ProbeConfig


```
{
  "probe_id": 0,
```


(次のページに続く)

(前のページからの続き)


```
  "probe_name": "string",
  "enabled": true,
  "interval": 0,
  "lifetime": 0,
  "default_enabled": true,
  "default_interval": 0,
  "default_lifetime": 0,
  "target_type": 0,
  "force_enabled": true
}
```


### プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| probe_id | integer | false | none | なし |
| probe_name | string | false | none | なし |
| enabled | boolean | false | none | なし |
| interval | integer | false | none | 実行頻度分。 |
| lifetime | integer | false | none | 履歴の保持日数。 |
| default_enabled | boolean | false | none | なし |
| default_interval | integer | false | none | なし |
| default_lifetime | integer | false | none | なし |
| target_type | integer | false | none | プローブのターゲットタイプ識別子たとえば、200=サーバー、100=エージェント。 |
| force_enabled | boolean | false | none | なし |

## ProbeConfigUpsert


```
{
  "probe_id": 7,
  "enabled": true,
  "enabled_by_default": true,
  "execution_frequency": 0,
  "lifetime": 0
}
```


プロファイルに追加または更新するプローブ構成フィールド。

プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| probe_id | integer | true | none | 追加/更新するプローブ識別子。 |
| enabled | boolean | false | none | プロファイルでこのプローブを有効/無効にします。 |
| enabled_by_default | boolean | false | none | プローブがデフォルトの有効化に従うかどうか。エイリアス 'use_default_enabled' が受け入れられます。 |
| execution_frequency | integer | false | none | 分単位のプローブ実行頻度。エイリアス'interval' が受け入れられます。 |
| lifetime | integer | false | none | 履歴の保持日数。 |

### ProbeConfigsUpdateOnly

```
{
  "changed": [
    {
      "probe_id": 7,
      "enabled": true,
      "enabled_by_default": true,
      "execution_frequency": 0,
      "lifetime": 0
    }
  ]
}
```

プロファイル ID で更新する場合のプローブ構成のデルタ’changed’を使用して既存のプローブのみを変更します。このエンドポイントを介したプローブの追加はサポートされていません。

プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| changed | [ProbeConfigUpsert ] | false | none | 変更する既存のプローブ構成 probe_id で識別されます。 |

### AlertChange

```
{
  "id": 0,
  "name": "string",
  "enabled": true,
  "history_retention": 0,
  "frequency_min": 0,
  "operator": ">",
  "low_threshold_value": 0,
  "medium_threshold_value": 0,
  "high_threshold_value": 0,
  "all_alert_enable": true,
  "email_group_id": 0,
  "low_alert_enable": true,
  "low_email_group_id": 0,
  "med_alert_enable": true,
  "med_email_group_id": 0,
  "high_alert_enable": true,
  "high_email_group_id": 0
}
```

プロファイルの既存のアラート構成を変更します。

#### プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| id | integer | false | none | 変更するアラート ID。 |
| name | string | false | none | アラート名。 |
| enabled | boolean | false | none | なし |
| history_retention | integer | false | none | なし |
| frequency_min | integer | false | none | なし |
| operator | string | false | none | なし |
| low_threshold_value | number | false | none | なし |
| medium_threshold_value | number | false | none | なし |
| high_threshold_value | number | false | none | なし |
| all_alert_enable | boolean | false | none | なし |
| email_group_id | integer | false | none | なし |
| low_alert_enable | boolean | false | none | なし |
| low_email_group_id | integer | false | none | なし |
| med_alert_enable | boolean | false | none | なし |
| med_email_group_id | integer | false | none | なし |
| high_alert_enable | boolean | false | none | なし |

次のページに続く

表 576 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| high_email_group_id | integer | false | none | なし |

### 列挙値

| Property | Value |
|---|---|
| operator | > |
| operator | < |

## AlertDeleteRef

```
{
  "id": 0
}
```

削除する既存のアラートへの参照 ID のみ。

### プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| id | integer | true | none | 削除するアラート構成 ID。 |

## AlertConfigsDelta

```
{
  "changed": [
    {
      "id": 0,
      "name": "string",
      "enabled": true,
      "history_retention": 0,
      "frequency_min": 0,
      "operator": ">",
      "low_threshold_value": 0,
      "medium_threshold_value": 0,
      "high_threshold_value": 0,
      "all_alert_enable": true,
      "email_group_id": 0,
      "low_alert_enable": true,
      "low_email_group_id": 0,
```

(次のページに続く)

(前のページからの続き)

```
      "med_alert_enable": true,
      "med_email_group_id": 0,
      "high_alert_enable": true,
      "high_email_group_id": 0
    }
  ],
  "added": [
    {
      "alert_name": "string",
      "alert_template": 0,
      "description": "string",
      "enabled": true,
      "history_retention": 0,
      "frequency_min": 0,
      "operator": "string",
      "low_threshold_value": 0,
      "medium_threshold_value": 0,
      "high_threshold_value": 0,
      "all_alert_enable": true,
      "email_group_id": 1,
      "low_alert_enable": true,
      "low_email_group_id": 1,
      "med_alert_enable": true,
      "med_email_group_id": 1,
      "high_alert_enable": true,
      "high_email_group_id": 1,
      "send_trap": false,
      "snmp_trap_version": 0,
      "low_send_trap": true,
      "med_send_trap": true,
      "high_send_trap": true,
      "cleared_alert_enable": true,
      "params": [],
      "execute_script": true,
      "execute_script_on_clear": true,
      "execute_script_on_pem_server": true,
      "script_code": "string",
      "submit_to_nagios": true,
      "override_default_config": true,
      "send_notification": false,
      "low_webhook_ids": [],
      "med_webhook_ids": [],
      "high_webhook_ids": [],
      "cleared_webhook_ids": []
    }
```

(次のページに続く)

(前のページからの続き)


```
  ],
  "deleted": [
    {
      "id": 0
    }
  ]
}
```


プロファイル内のアラート構成のデルタ。

### プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| changed | [AlertChange ] | false | none | 既存のアラートを変更します。 |
| added | [AlertPayload ] | false | none | 新しいアラートを追加します。必須フィールドについては、AlertPayload を参照してください。 |
| deleted | [AlertDeleteRef ] | false | none | ID ごとに既存のアラートを削除します。 |

## DeleteAndReassignPayload


```
{
  "old_profile_id": 42,
  "new_profile_id": 84
}
```


### プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| old_profile_id | integer | true | none | 削除する既存のプロファイル ID。 |
| new_profile_id | integer | false | none | 関連するサーバー/エージェントを再割り当てするための新しいプロファイル ID オプショナル。 |

### プロファイル割り当ての詳細


```
{
  "has_profile": true,
  "profile_id": 0,
  "profile_name": "string"
}
```


### プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| has_profile | boolean | false | none | なし |
| profile_id | integer | false | none | なし |
| profile_name | string | false | none | なし |

## AlertPayload


```
{
  "alert_name": "string",
  "alert_template": 0,
  "description": "string",
  "enabled": true,
  "history_retention": 0,
  "frequency_min": 0,
  "operator": "string",
  "low_threshold_value": 0,
  "medium_threshold_value": 0,
  "high_threshold_value": 0,
  "all_alert_enable": true,
  "email_group_id": 1,
  "low_alert_enable": true,
  "low_email_group_id": 1,
  "med_alert_enable": true,
  "med_email_group_id": 1,
  "high_alert_enable": true,
  "high_email_group_id": 1,
  "send_trap": false,
  "snmp_trap_version": 0,
  "low_send_trap": true,
  "med_send_trap": true,
  "high_send_trap": true,
  "cleared_alert_enable": true,
```


(次のページに続く)

(前のページからの続き)


```
  "params": [],
  "execute_script": true,
  "execute_script_on_clear": true,
  "execute_script_on_pem_server": true,
  "script_code": "string",
  "submit_to_nagios": true,
  "override_default_config": true,
  "send_notification": false,
  "low_webhook_ids": [],
  "med_webhook_ids": [],
  "high_webhook_ids": [],
  "cleared_webhook_ids": []
}
```


### プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| alert_name | string | false | none | アラートの名前。注 このプロパティは、POST 要求の必須パラメーターです。 |
| alert_template | integer | false | none | アラートテンプレートの ID。 ID は正の整数値である必要があります。注 このプロパティは、POST 要求の必須パラメーターです。 |
| description | string | false | none | なし |
| enabled | boolean | false | none | このアラートを有効/無効にするかどうかを定義します。注 このプロパティは、POST 要求の必須パラメーターです。 |
| history_retention | integer | false | none | このアラートの履歴を保持する日数を指定します。指定できる値は 1 ～99999 です。 注このプロパティは、POST 要求の必須パラメーターです。 |

次のページに続く

表 582 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| frequency_min | integer | false | none | PEM データベースサーバーでアラートを確認する頻度を分単位で定義します。指定できる値は 1 〜65534 です。 注このプロパティは、POST 要求の必須パラメーターです。 |
| operator | string | false | none | しきい値をチェックするための演算子を定義します。有効な文字列は>"および"<"です。注このプロパティは、POST 要求の必須パラメーターです。 |
| low_threshold_value | number | false | none | 低アラートをトリガーするためのしきい値。このプロパティには、有効な整数/浮動小数点値を指定できます。注これは POST 要求の必須パラメーターです。 |
| medium_threshold_value | number | false | none | 中アラートをトリガーするためのしきい値。このプロパティには、有効な整数/浮動小数点値を指定できます。注これは POST 要求の必須パラメーターです。 |
| high_threshold_value | number | false | none | 高度なアラートをトリガーするためのしきい値。このプロパティには、有効な整数/浮動小数点値を指定できます。注これは POST 要求の必須パラメーターです。 |
| all_alert_enable | boolean | false | none | アラートの状態が変更されたときに、パラメーター 'email_group_id' で指定されたグループにメール通知を送信するかどうかを決定します。 |

次のページに続く

表 582 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| email_group_id | integer | false | none | アラートの状態が変更されたときに、メール通知の送信先のメールグループを定義します。 |
| low_alert_enable | boolean | false | none | アラートの状態が LOW に設定されているときに、パラメーター 'low_email_group_id' で指定されたグループにメール通知を送信するかどうかを決定します。 |
| low_email_group_id | integer | false | none | アラートの状態が LOW に変更されたときにメール通知の送信先の電子メールグループを定義します。 |
| med_alert_enable | boolean | false | none | アラートの状態が MEDIUM に設定されている場合、パラメーター 'medium_email_group_id' で指定されたグループにメール通知を送信するかどうかを決定します。 |
| med_email_group_id | integer | false | none | アラートの状態が MEDIUM に変更されたときにメール通知の送信先のメールグループを定義します。 |
| high_alert_enable | boolean | false | none | アラートの状態が HIGH に設定されているときに、パラメーター 'high_email_group_id' で指定されたグループに電子メール通知を送信するかどうかを決定します。 |
| high_email_group_id | integer | false | none | アラートの状態が HIGH に変更されたときに電子メール通知の送信先の電子メールグループを定義します。 |

次のページに続く

表 582 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| send_trap | boolean | false | none | アラートの状態が変更されたときに SNMP トラップ通知を送信するかどうかを決定します。 |
| snmp_trap_version | integer | false | none | SNMP トラップバージョンを定義します。サポートされている SNMP トラップバージョンは 1、2、または 3 です。 |
| low_send_trap | boolean | false | none | アラートの状態が LOW に変更されたときに SNMP トラップ通知を送信するかどうかを決定します。 |
| med_send_trap | boolean | false | none | アラートの状態が MEDIUM に変更されたときに SNMP トラップ通知を送信するかどうかを決定します。 |
| high_send_trap | boolean | false | none | アラートの状態が HIGH に変化したときに SNMP トラップ通知を送信するかどうかを決定します。 |
| cleared_alert_enable | boolean | false | none | 'クリアアラート' の通知をミュートするかどうかを決定します。 |
| params | [AlertParams ] | false | none | アラートを確認するためにアラートテンプレートが使用するパラメーター/値のペアのリスト。注アラートテンプレートには、all パラメーターを定義する必要があります。 |
| execute_script | boolean | false | none | アラートの状態が LOW/MEDIUM/HIGH に変更された場合に、提供されたスクリプトを実行するかどうかを決定します。 |

次のページに続く

表 582 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| execute_script_on_clear | boolean | false | none | アラートの状態がクリアされた場合、提供されたスクリプトを実行するかどうかを決定します。 |
| execute_script_on_pem_serve | boolean | false | none | 提供されたスクリプトを PEM ホストまたはエージェントがオブジェクトを監視しているホストで実行するかどうかを決定します。 |
| script_code | string | false | none | 実行するシェルまたはバッチスクリプト。詳細については、ドキュメントをご覧ください。 |
| submit_to_nagios | boolean | false | none | アラートの状態が変更されたときに、nagios に通知を送信するかどうかを定義します。 |
| override_default_config | boolean | false | none | デフォルトの Webhook 構成をオーバーライドするかどうかを決定します。 |
| send_notification | boolean | false | none | Webhook 送信通知を有効にするかどうかを決定します。 |
| low_webhook_ids | [integer] | false | none | 低アラートを構成できる Webhook ID を定義します。 |
| med_webhook_ids | [integer] | false | none | 中程度のアラートを構成できる Webhook ID を定義します。 |
| high_webhook_ids | [integer] | false | none | 高アラートを構成できる Webhook ID を定義します。 |
| cleared_webhook_ids | [integer] | false | none | クリアされたアラートを構成できる Webhook ID を定義します。 |

### アラート

```
{
  "id": 0,
  "alert_name": "string",
```

(次のページに続く)

(前のページからの続き)


```
  "alert_template": 0,
  "description": "string",
  "enabled": true,
  "history_retention": 0,
  "frequency_min": 0,
  "auto_created": true,
  "template_id": "string",
  "params": [],
  "param_names": ["string"],
  "param_types": "string",
  "params_units": ["string"],
  "threshold_unit": "string",
  "operator": "string",
  "low_threshold_value": 0,
  "medium_threshold_value": 0,
  "high_threshold_value": 0,
  "send_email": true,
  "all_alert_enable": true,
  "email_group_id": 0,
  "low_alert_enable": true,
  "low_email_group_id": 0,
  "med_alert_enable": true,
  "med_email_group_id": 0,
  "high_alert_enable": true,
  "high_email_group_id": 0,
  "send_trap": true,
  "snmp_trap_version": 0,
  "low_send_trap": true,
  "med_send_trap": true,
  "high_send_trap": true,
  "cleared_alert_enable": true,
  "execute_script": true,
  "execute_script_on_clear": true,
  "execute_script_on_pem_server": true,
  "script_code": "string",
  "submit_to_nagios": true,
  "override_default_config": true,
  "send_notification": true,
  "low_webhook_ids": [0],
  "med_webhook_ids": [0],
  "high_webhook_ids": [0],
  "cleared_webhook_ids": [0]
}
```

プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| id | integer | false | none | アラート ID。注このプロパティは、POST 要求には適用されません。 |
| alert_name | string | false | none | アラートの名前。注 このプロパティは、POST 要求の必須パラメーターです。 |
| alert_template | integer | false | none | アラートテンプレートの ID。このプロパティは、正の整数値である必要があります。注これはPOST 要求の必須パラメーターです。 |
| description | string | false | none | なし |
| enabled | boolean | false | none | このアラートが有効/無効かどうかを判断します。注 このプロパティは、POST 要求の必須パラメーターです。 |
| history_retention | integer | false | none | このアラートの履歴を保持する日数を指定します。指定できる値は 1 〜99999 です。 注このプロパティは、POST 要求の必須パラメーターです。 |
| frequency_min | integer | false | none | PEM データベースサーバーでアラートを確認する頻度を分単位で定義します。指定できる値は 1 〜65534 です。 注このプロパティは、POST 要求の必須パラメーターです。 |
| auto_created | boolean | false | none | アラートを自動作成するかどうかを指定します。 |
| template_id | string | false | none | アラートテンプレートを定義します。 |

次のページに続く

表 583 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| params | [AlertParams ] | false | none | アラートを確認するためにアラートテンプレートが使用するパラメーター/値のペアのリスト。注アラートテンプレートには、all パラメーターを定義する必要があります。 |
| param_names | [string] | false | none | パラメーターの名前を定義します。 |
| param_types | string | false | none | パラメーターのタイプを定義します。例 INTEGER/STRING.. |
| params_units | [string] | false | none | パラメーターに使用される単位を定義します。例 'DAYS'、'True or False'、'Hours'、'Minutes' など |
| threshold_unit | string | false | none | MB/kB/秒/日/... などの、しきい値の計算に使用される単位 この値は、GUI でのみ使用されます。 |
| operator | string | false | none | しきい値をチェックするための演算子を定義します。有効な文字列は>"および"<"です。注 このプロパティは、POST 要求の必須パラメーターです。 |
| low_threshold_value | number | false | none | 低アラートをトリガーするためのしきい値。このプロパティには、有効な整数/浮動小数点値を指定できます。注これは POST 要求の必須パラメーターです。 |
| medium_threshold_value | number | false | none | 中アラートをトリガーするためのしきい値。このプロパティには、有効な整数/浮動小数点値を指定できます。注これは POST 要求の必須パラメーターです。 |

次のページに続く

表 583 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| high_threshold_value | number | false | none | 高度なアラートをトリガーするためのしきい値。任意の有効な整数/浮動小数点値を指定できます。注これはPOST要求の必須パラメーターです。 |
| send_email | boolean | false | none | アラートの状態が変更されたときに電子メールを送信するかどうかを決定します。 |
| all_alert_enable | boolean | false | none | アラートの状態が変更されたときに、パラメーター 'email_group_id' で指定されたグループにメール通知を送信するかどうかを決定します。 |
| email_group_id | integer | false | none | アラートの状態が変更されたときに電子メール通知の送信先の電子メールグループを定義します。 |
| low_alert_enable | boolean | false | none | アラートの状態がLOWに設定されているときに、パラメーター 'low_email_group_id' で指定されたグループにメール通知を送信するかどうかを決定します。 |
| low_email_group_id | integer | false | none | アラートの状態がLOWに変更されたときに電子メール通知の送信先の電子メールグループを定義します。 |
| med_alert_enable | boolean | false | none | アラートの状態がMEDIUMに設定されている場合、パラメーター 'medium_email_group_id' で指定されたグループにメール通知を送信するかどうかを決定します。 |

次のページに続く

表 583 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| med_email_group_id | integer | false | none | アラートの状態が MEDIUM に変更されたときにメール通知の送信先のメールグループを定義します。 |
| high_alert_enable | boolean | false | none | アラートの状態が HIGH に設定されているときに、パラメーター 'high_email_group_id' で指定されたグループに電子メール通知を送信するかどうかを決定します。 |
| high_email_group_id | integer | false | none | アラートの状態が HIGH に変更されたときに電子メール通知の送信先の電子メールグループを定義します。 |
| send_trap | boolean | false | none | アラートの状態が変更されたときに SNMP トラップ通知を送信するかどうかを決定します。 |
| snmp_trap_version | integer | false | none | SNMP トラップバージョンを定義します。サポートされている SNMP トラップバージョンは 1、2、または 3 です |
| low_send_trap | boolean | false | none | アラートの状態が LOW に変更されたときに SNMP トラップ通知を送信するかどうかを決定します。 |
| med_send_trap | boolean | false | none | アラートの状態が MEDIUM に変更されたときに SNMP トラップ通知を送信するかどうかを決定します。 |
| high_send_trap | boolean | false | none | アラートの状態が HIGH に変わったときに SNMP トラップ通知を送信するかどうかを決定します。 |

次のページに続く

表 583 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| cleared_alert_enable | boolean | false | none | 'クリアアラート' の通知をミュートするかどうかを決定します。 |
| execute_script | boolean | false | none | アラートの状態が LOW/MEDIUM/HIGH に変更された場合に、提供されたスクリプトを実行するかどうかを決定します。 |
| execute_script_on_clear | boolean | false | none | アラートの状態がクリアされた場合に、提供されたスクリプトを実行するかどうかを決定します。 |
| execute_script_on_pem_serve | boolean | false | none | 提供されたスクリプトを PEM ホストまたはエージェントがオブジェクトを監視しているホストで実行するかどうかを決定します。 |
| script_code | string | false | none | 実行するシェルまたはバッチスクリプト。詳細については、ドキュメントをご覧ください。 |
| submit_to_nagios | boolean | false | none | アラートの状態が変更されたときに nagios に通知を送信するかどうかを決定します。 |
| override_default_config | boolean | false | none | デフォルトの Webhook 構成をオーバーライドするかどうかを決定します。 |
| send_notification | boolean | false | none | Webhook 送信通知を有効にするかどうかを決定します。 |
| low_webhook_ids | [integer] | false | none | 低アラートが構成されている Webhook ID |
| med_webhook_ids | [integer] | false | none | 中アラートが構成されている Webhook ID |
| high_webhook_ids | [integer] | false | none | 高アラートが構成されている Webhook ID |
| cleared_webhook_ids | [integer] | false | none | クリアされたアラートが構成されている Webhook ID |

### AlertParams

```
{
  "paramname": "Exclude tables smaller than",
  "paramvalue": "string"
}
```

#### プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| paramname | string | false | none | パラメーターの名前。注このプロパティは、アラートテンプレートパラメーターリストに存在する必要があります。例 |
| paramvalue | string | false | none | 指定されたパラメーターの値 |

#### 列挙値

| Property | Value |
|---|---|
| paramname | Exclude tables smaller than |
| paramname | mount point |
| paramname | Host IP Address |
| paramname | Time interval to monitor |
| paramname | Time since idle |
| paramname | Per CPU utilization (K%) |

### AlertTemplateParams

```
{
  "param_names": "string",
  "param_types": "string",
  "params_units": "string"
}
```

#### プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| param_names | string | false | none | パラメーターの名前を定義します。 |

次のページに続く

表 586 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| param_types | string | false | none | パラメーターのタイプを定義します。例 INTEGER/STRING.. |
| params_units | string | false | none | パラメーターに使用される単位を定義します。例 'DAYS'、'True or False'、'Hours'、'Minutes' など |

### ProbeDependencyParams

```
{
  "internal_name": "string"
}
```

プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| internal_name | string | false | none | プローブの内部名。 |

### AlternateServerProbeCode

```
{
  "server_version_id": 0,
  "server_probe_code": "string",
  "extension_version": 0
}
```

プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| server_version_id | integer | false | none | 監視対象サーバーのサーバーバージョン ID<table class="model"><thead><tr><th>server_versio Server</th></tr></thead><thead /> <tbody><tr><td>10905</td><td>PostgreSQL 9.5</td></tr><tr><td>10906</td><td>Postgre 9.6</td></tr ><tr><td>11000</td><td>PostgreSQL 10</td></tr><tr><td>11100</td><td>Postgre 11</td></tr< tr><td>11200</td><td>PostgreSQL 12</td></tr><tr><td>11300</td><td>Postgre 13</td></tr><tr> <td>20905</td><td>Postgres Plus Advanced Server 9.5</td></tr><tr><td>20906</td><td>Postgre Plus Advanced Server 9.6</td></tr ><tr><td>21000</td><td> EDB Postgres Advanced Server 10</td></tr><tr><td>21100</td><td> EDB Postgres Advanced Server 11</td ></tr><tr><td>21200 </td><td> EDB Postgres Advanced Server 12</td></tr><tr><td>21300 </td><td> EDB Postgres Advanced Server 13</td></tr></tbody></table> |
| server_probe_code | string | false | none | 指定されたサーバーバージョンで実行する SQL クエリー |
| extension_version | integer | false | none | 拡張機能のバージョンを指定します。拡張ターゲットタイプに有効な |

### DeletedAlternateServerProbeCode

```
{
  "server_version_id": 0
}
```

#### プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| server_version_id | integer | false | none | 監視対象サーバーのサーバーバージョン ID<table class="model"><thead><tr><th>server_version Server</th></tr></thead><thead /> <tbody><tr><td>10905</td><td>PostgreSQL 9.5</td></tr><tr><td>10906</td><td>PostgreS 9.6</td></tr ><tr><td>11000</td><td>PostgreSQL 10</td></tr><tr><td>11100</td><td>PostgreS 11</td></tr>< tr><td>11200</td><td>PostgreSQL 12</td></tr><tr><td>11300</td><td>PostgreS 13</td></tr><tr> <td>20905</td><td>Postgres Plus Advanced Server 9.5</td></tr><tr><td>20906</td><td>Postgres Plus Advanced Server 9.6</td></tr ><tr><td>21000</td><td> EDB Postgres Advanced Server 10</td></tr><tr><td>21100</td><td> EDB Postgres Advanced Server 11</td ></tr><tr><td>21200 </td><td> EDB Postgres Advanced Server 12</td></tr><tr><td>21300 </td><td> EDB Postgres Advanced Server 13</td></tr></tbody></table> |

プローブカラム


```
{
  "pc_id": 0,
  "pc_unit": "string",
  "pc_graphable": true
}
```


プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| pc_id | integer | false | none | 列識別子 |
| pc_unit | string | false | none | 列の単位を返します。 |
| pc_graphable | boolean | false | none | 列のデータをチャートでグラフ化できるかどうかを決定します。 |

### ChartExportTemplate


```
{
  "chart_title": "string",
  "chart_type": "string",
  "chart_category": "string",
  "chart_description": "string",
  "reference_id": "string",
  "chart_level": 0,
  "chart_line_ext": [0],
  "chart_line_ext_metric": "string",
  "chart_line_ext_metric_options": [0],
  "chart_line_ext_opt": "string",
  "chart_line_ext_val": "string",
  "chart_line_extrapolated_type": "string",
  "chart_line_points": 0,
  "chart_line_span": [0],
  "chart_refresh": 0,
  "espan": 0,
  "line_span": 0,
  "sel_metrics_L": [{}],
  "sel_metrics_T": [{}],
  "sel_metrics_C": [{}],
  "shared": [0],
  "shared_all": true
}
```

プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| chart_title | string | false | none | チャートのタイトル。注 このプロパティは、POST 要求の必須パラメーターです。 |
| chart_type | string | false | none | チャートカテゴリ。注 このプロパティは、POST 要求の必須パラメーターです。 |
| chart_category | string | false | none | チャートタイプ。注 このプロパティは、POST 要求の必須パラメーターです。 |
| chart_description | string | false | none | チャートの説明。注 このプロパティは、POST 要求の必須パラメーターです。 |
| reference_id | string | false | none | チャートの内部識別子。 |
| chart_level | integer | false | none | チャートレベルを定義します。注 このプロパティは、POST 要求の必須パラメーターです。 |
| chart_line_ext | [integer] | false | none | 折れ線グラフの外挿を定義します。注 このプロパティは、POST 要求の必須パラメーターです。 |
| chart_line_ext_metric | string | false | none | 折れ線グラフの外挿メトリックを定義します。注 このプロパティは、POST 要求の必須パラメーターです。 |
| chart_line_ext_metric_opti | [integer] | false | none | 折れ線グラフの外挿メトリックオプションを定義します。注 このプロパティは、POST 要求の必須パラメーターです。 |
| chart_line_ext_opt | string | false | none | 折れ線グラフの外挿オプションを定義します。注 このプロパティは、POST 要求の必須パラメーターです。 |

次のページに続く

表 591 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| chart_line_ext_val | string | false | none | 折れ線グラフの外挿値を定義します。注 このプロパティは、POST 要求の必須パラメーターです。 |
| chart_line_extrapolated_ty | string | false | none | 折れ線グラフの外挿タイプを定義します。注 このプロパティは、POST 要求の必須パラメーターです。 |
| chart_line_points | integer | false | none | 折れ線グラフの外挿ポイントを定義します。 |
| chart_line_span | [integer] | false | none | 折れ線グラフのスパンを定義します。注 このプロパティは、POST 要求の必須パラメーターです。 |
| chart_refresh | integer | false | none | チャートの更新時間 |
| espan | integer | false | none | チャートスパン |
| line_span | integer | false | none | チャートラインスパン |
| sel_metrics_L | [object] | false | none | 折れ線グラフメトリックとそのプローブデータを定義します。 |
| sel_metrics_T | [object] | false | none | テーブルチャートメトリックとそのプローブデータを定義します。 |
| sel_metrics_C | [object] | false | none | キャパシティマネージャーチャートメトリックとそのプローブデータを定義します。 |
| shared | [integer] | false | none | チャートが共有されたロール ID を定義します。 |
| shared_all | boolean | false | none | チャートをすべての人と共有する必要があるかどうかを決定します。 |

### DashboardExportTemplate

```
{
  "name": "string",
  "descp": "string",
  "reference_id": "string",
  "font": "string",
```

(次のページに続く)

(前のページからの続き)

```
  "font_size": 0,
  "level": 0,
  "is_ops": true,
  "show_title": true,
  "design_layout": [{}],
  "c_charts": [{}],
  "shared": [0],
  "shared_all": true
}
```

### プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| name | string | false | none | ダッシュボード名を定義します。注 このプロパティは、POST 要求の必須パラメーターです。 |
| descp | string | false | none | ダッシュボードの説明を提供します。 |
| reference_id | string | false | none | ダッシュボードの内部識別子 |
| font | string | false | none | ダッシュボードのフォントを定義します。 |
| font_size | integer | false | none | ダッシュボードのフォントサイズを定義します。 |
| level | integer | false | none | ダッシュボードレベルを定義します。 |
| is_ops | boolean | false | none | ダッシュボードが ops 用であるかどうかを判断します。 |
| show_title | boolean | false | none | ダッシュボードのタイトルを表示するかどうかを決定します。 |
| design_layout | [object] | false | none | ダッシュボードのデザインレイアウトデータを定義します。 |
| c_charts | [object] | false | none | 依存するカスタムチャートとそのプローブデータを定義します。 |
| shared | [integer] | false | none | ダッシュボードを共有するロール ID を定義します。 |

次のページに続く

表 592 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| shared_all | boolean | false | none | ダッシュボードをすべてのユーザーと共有する必要があるかどうかを決定します |

### CustomDashboardExportData

```
{
  "dashboards": [0]
}
```

### プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| dashboards | [integer] | false | none | ダッシュボード ID のリスト |

### CustomDashboardExportResponseData

```
{
  "version": 1,
  "dashboards": [
    {
      "name": "string",
      "descp": "string",
      "reference_id": "string",
      "font": "string",
      "font_size": 0,
      "level": 0,
      "is_ops": true,
      "show_title": true,
      "design_layout": [{}],
      "c_charts": [{}],
      "shared": [0],
      "shared_all": true
    }
  ]
}
```

プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| version | integer | false | none | ダッシュボードエクスポートのバージョン |
| dashboards | [DashboardExportTemplate] | false | none | なし |

### CustomDashboardImportPostData

```
{
  "version": 1,
  "skip_overwrite": true,
  "skip_overwrite_probe": true,
  "skip_overwrite_chart": true,
  "dashboards": [
    {
      "name": "string",
      "descp": "string",
      "reference_id": "string",
      "font": "string",
      "font_size": 0,
      "level": 0,
      "is_ops": true,
      "show_title": true,
      "design_layout": [{}],
      "c_charts": [{}],
      "shared": [0],
      "shared_all": true
    }
  ]
}
```

プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| version | integer | false | none | ダッシュボードエクスポートのバージョン |
| skip_overwrite | boolean | false | none | 既存のチャートをスキップできるかどうかを決定します。 |

次のページに続く

表 595 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| skip_overwrite_probe | boolean | false | none | 既存の依存プローブをスキップできるかどうかを決定します。 |
| skip_overwrite_chart | boolean | false | none | 既存の依存チャートをスキップできるかどうかを決定します。 |
| dashboards | [DashboardExportTemplate] | false | none | なし |

### CustomChartExportData

```
{
  "charts": [0]
}
```

#### プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| charts | [integer] | false | none | チャート ID のリスト |

### CustomChartExportResponseData

```
{
  "version": 1,
  "charts": [
    {
      "chart_title": "string",
      "chart_type": "string",
      "chart_category": "string",
      "chart_description": "string",
      "reference_id": "string",
      "chart_level": 0,
      "chart_line_ext": [0],
      "chart_line_ext_metric": "string",
      "chart_line_ext_metric_options": [0],
      "chart_line_ext_opt": "string",
      "chart_line_ext_val": "string",
      "chart_line_extrapolated_type": "string",
      "chart_line_points": 0,
      "chart_line_span": [0],
```

(次のページに続く)

(前のページからの続き)

```
      "chart_refresh": 0,
      "espan": 0,
      "line_span": 0,
      "sel_metrics_L": [{}],
      "sel_metrics_T": [{}],
      "sel_metrics_C": [{}],
      "shared": [0],
      "shared_all": true
    }
  ]
}
```

### プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| version | integer | false | none | チャートエクスポートのバージョン |
| charts | [ChartExportTemplate] | false | none | なし |

### CustomChartImportPostData

```
{
  "version": 1,
  "skip_overwrite": true,
  "skip_overwrite_probe": true,
  "charts": [
    {
      "chart_title": "string",
      "chart_type": "string",
      "chart_category": "string",
      "chart_description": "string",
      "reference_id": "string",
      "chart_level": 0,
      "chart_line_ext": [0],
      "chart_line_ext_metric": "string",
      "chart_line_ext_metric_options": [0],
      "chart_line_ext_opt": "string",
      "chart_line_ext_val": "string",
      "chart_line_extrapolated_type": "string",
      "chart_line_points": 0,
      "chart_line_span": [0],
```

(次のページに続く)

(前のページからの続き)

```
      "chart_refresh": 0,
      "espan": 0,
      "line_span": 0,
      "sel_metrics_L": [{}],
      "sel_metrics_T": [{}],
      "sel_metrics_C": [{}],
      "shared": [0],
      "shared_all": true
    }
  ]
}
```

### プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| version | integer | false | none | チャートエクスポートのバージョン |
| skip_overwrite | boolean | false | none | 既存のチャートをスキップできるかどうかを決定します。 |
| skip_overwrite_probe | boolean | false | none | 既存の依存プローブをスキップできるかどうかを決定します。 |
| charts | [ChartExportTemplate] | false | none | なし |

## AlertImportExportTemplate

```
{
  "name": "string",
  "description": "string",
  "reference_id": "string",
  "default_history_retention": 0,
  "object_type": 0,
  "sql": "string",
  "applicable_on_server": "ALL",
  "default_check_frequency": 0,
  "threshold_unit": "string",
  "is_system_template": true,
  "is_auto_create": true,
  "thresholds": [0],
  "operator": ">",
```

(次のページに続く)

(前のページからの続き)


```
  "high_threshold_value": "string",
  "medium_threshold_value": "string",
  "low_threshold_value": "string",
  "info_sql": "string",
  "probes": [
    {
      "probe_name": "string",
      "collection_method": "w",
      "enabled": true,
      "target_type": 0,
      "any_server_version": true,
      "any_extension_version": true,
      "extension_name": "string",
      "interval": 0,
      "lifetime": 0,
      "discard_history": true,
      "platform": "unix",
      "probe_code": "string",
      "probe_columns": [
        {
          "pc_name": "string",
          "pc_data_type": "bigint",
          "pc_unit": "string",
          "pc_col_type": "m",
          "pc_graphable": true,
          "pc_calc_pit": true,
          "pc_pit_default": true
        }
      ],
      "alternate_code": [
        {
          "server_version_id": 0,
          "server_probe_code": "string",
          "extension_version": 0
        }
      ],
      "internal_name": "string"
    }
  ]
}
```

プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| name | string | false | none | アラートテンプレートの名前。注 このプロパティは、POST 要求の必須パラメーターです。 |
| description | string | false | none | アラートテンプレートの説明。注 このプロパティは、POST 要求の必須パラメーターです。 |
| reference_id | string | false | none | アラートテンプレートの内部識別子 |
| default_history_retention | integer | false | none | このタイプのアラートの履歴保持期間を日単位で定義します。注 このプロパティは、POST 要求の必須パラメーターです。 |

次のページに続く

表 599 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| object_type | integer | false | none | このアラートテンプレートを適用するオブジェクトのタイプを定義します。<ul><li>システム全体のアラートテンプレートは50です。</li><li>エージェントに適用されるアラートテンプレートは100です。</li> <li>管理対象サーバーに適用されるアラートテンプレートは200です。</li><li>特定の監視対象サーバーのデータベースに適用されるアラートテンプレートは300です。</li><li>管理対象サーバーの特定のデータベースのスキーマに適用されるアラートテンプレートは400です。</li><li>特定のスキーマのテーブルに適用されるアラートテンプレート</li><li>監視対象サーバー上のデータベースの特定のスキーマのインデックスに適用されるアラートテンプレートは600です。>.</li><li>監視対象サーバー上のデータベースの特定のスキーマのシーケンスに適用できるアラートテンプレートは700です。</li><li>に適用できるアラートテンプレート監視対象サーバー上のデータベースの特定のスキーマのファンクションは800で |

表 599 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| sql | string | false | none | しきい値に到達したことを示す数値を取得するための SQL クエリー。パラメーター化される場合があります。注 このプロパティは、POST 要求の必須パラメーターです。 |
| applicable_on_server | string | false | none | テンプレートを適用する監視対象サーバーの種類を指定します。使用可能な値は、ALL、POSTGRES_SERVER、ADVANCED_SERVER です。注 このプロパティは、POST 要求の必須パラメーターです。 |
| default_check_frequency | integer | false | none | このテンプレートが定義するアラートをチェックする頻度分単位を指定します。注 このプロパティは、POST 要求の必須パラメーターです。 |
| threshold_unit | string | false | none | MB/kB/秒/日/... などの、しきい値の計算に使用される単位 この値は、GUI でのみ使用されます。 |
| is_system_template | boolean | false | none | 指定されたテンプレートがシステム定義であるかユーザー定義であるかを判断します。注このプロパティは、POST/PUT 要求には適用されません。 |
| is_auto_create | boolean | false | none | エージェントまたはサーバーが PEM に追加されたらすぐにアラートを作成するかどうかを決定します。 |

次のページに続く

表 599 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| thresholds | [number] | false | none | デフォルトのしきい値。しきい値配列の値は、演算子に基づいて並べ替えられた順序である必要があります。 |
| operator | string | false | none | しきい値を比較する演算子 |
| high_threshold_value | string | false | none | 高しきい値 |
| medium_threshold_value | string | false | none | 中しきい値 |
| low_threshold_value | string | false | none | 低しきい値 |
| info_sql | string | false | none | ダッシュボードのアラートに関する詳細情報を提供する SQL クエリーを定義します。 |
| probes | [CustomProbeImportPostData ] | false | none | なし |

### 列挙値

| Property | Value |
|---|---|
| applicable_on_server | ALL |
| applicable_on_server | POSTGRES_SERVER |
| applicable_on_server | ADVANCED_SERVER |
| operator | > |
| operator | < |

## CustomAlertTemplateExportData

```
{
  "alert_templates": [0]
}
```

### プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| alert_templates | [integer] | false | none | アラートテンプレート ID のリスト |

### CustomAlertTemplateExportResponseData


```
{
  "version": 1,
  "alert_templates": [
    {
      "name": "string",
      "description": "string",
      "reference_id": "string",
      "default_history_retention": 0,
      "object_type": 0,
      "sql": "string",
      "applicable_on_server": "ALL",
      "default_check_frequency": 0,
      "threshold_unit": "string",
      "is_system_template": true,
      "is_auto_create": true,
      "thresholds": [0],
      "operator": ">",
      "high_threshold_value": "string",
      "medium_threshold_value": "string",
      "low_threshold_value": "string",
      "info_sql": "string",
      "probes": [
        {
          "probe_name": "string",
          "collection_method": "w",
          "enabled": true,
          "target_type": 0,
          "any_server_version": true,
          "any_extension_version": true,
          "extension_name": "string",
          "interval": 0,
          "lifetime": 0,
          "discard_history": true,
          "platform": "unix",
          "probe_code": "string",
          "probe_columns": [
            {
              "pc_name": "string",
              "pc_data_type": "bigint",
              "pc_unit": "string",
              "pc_col_type": "m",
              "pc_graphable": true,
```


(次のページに続く)

(前のページからの続き)

```
            "pc_calc_pit": true,
            "pc_pit_default": true
          }
        ],
        "alternate_code": [
          {
            "server_version_id": 0,
            "server_probe_code": "string",
            "extension_version": 0
          }
        ],
        "internal_name": "string"
      }
    ]
  }
]
}
```

### プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| version | integer | false | none | アラートテンプレートのエクスポートのバージョン |
| alert_templates | [AlertImportExportTemplate ] | false | none | なし |

### CustomAlertTemplateImportPostData

```
{
  "version": 1,
  "skip_overwrite": true,
  "skip_overwrite_probe": true,
  "alert_templates": [
    {
      "name": "string",
      "description": "string",
      "reference_id": "string",
      "default_history_retention": 0,
      "object_type": 0,
      "sql": "string",
      "applicable_on_server": "ALL",
```

(次のページに続く)

(前のページからの続き)

```
    "default_check_frequency": 0,
    "threshold_unit": "string",
    "is_system_template": true,
    "is_auto_create": true,
    "thresholds": [0],
    "operator": ">",
    "high_threshold_value": "string",
    "medium_threshold_value": "string",
    "low_threshold_value": "string",
    "info_sql": "string",
    "probes": [
      {
        "probe_name": "string",
        "collection_method": "w",
        "enabled": true,
        "target_type": 0,
        "any_server_version": true,
        "any_extension_version": true,
        "extension_name": "string",
        "interval": 0,
        "lifetime": 0,
        "discard_history": true,
        "platform": "unix",
        "probe_code": "string",
        "probe_columns": [
          {
            "pc_name": "string",
            "pc_data_type": "bigint",
            "pc_unit": "string",
            "pc_col_type": "m",
            "pc_graphable": true,
            "pc_calc_pit": true,
            "pc_pit_default": true
          }
        ],
        "alternate_code": [
          {
            "server_version_id": 0,
            "server_probe_code": "string",
            "extension_version": 0
          }
        ],
        "internal_name": "string"
      }
    ]
```

(次のページに続く)

(前のページからの続き)

```
    }
  ]
}
```

### プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| version | integer | false | none | アラートテンプレートのエクスポートのバージョン |
| skip_overwrite | boolean | false | none | 既存のアラートテンプレートをスキップできるかどうかを決定します。 |
| skip_overwrite_probe | boolean | false | none | 既存の依存プローブをスキップできるかどうかを決定します。 |
| alert_templates | [AlertImportExportTemplate ] | false | none | なし |

### AlertTemplate

```
{
  "name": "string",
  "description": "string",
  "object_type": 0,
  "param_names": ["string"],
  "param_types": "string",
  "params_units": ["string"],
  "sql": "string",
  "applicable_on_server": "ALL",
  "default_check_frequency": 0,
  "default_history_retention": 0,
  "probe_dependency_list": ["string"],
  "threshold_unit": "string",
  "is_system_template": false,
  "is_auto_create": true,
  "operator": ">",
  "low_threshold_value": 0,
  "medium_threshold_value": 0,
  "high_threshold_value": 0,
  "info_sql": "string"
}
```

プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| name | string | false | none | アラートテンプレートの名前。注 このプロパティは、POST 要求の必須パラメーターです。 |
| description | string | false | none | アラートテンプレートの説明。注 このプロパティは、POST 要求の必須パラメーターです。 |

次のページに続く

表 604 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| object_type | integer | false | none | このアラートテンプレートを適用するオブジェクトのタイプを定義します。<ul><li>システム全体のアラートテンプレートは50です。</li><li>エージェントに適用されるアラートテンプレートは100です。</li> <li>管理対象サーバーに適用されるアラートテンプレートは200です。</li><li>特定の監視対象サーバーのデータベースに適用されるアラートテンプレートは300です。</li><li>管理対象サーバーの特定のデータベースのスキーマに適用されるアラートテンプレートは400です。</li><li>特定のスキーマのテーブルに適用されるアラートテンプレート</li><li>監視対象サーバー上のデータベースの特定のスキーマのインデックスに適用されるアラートテンプレートは600です。>.</li><li>監視対象サーバー上のデータベースの特定のスキーマのシーケンスに適用できるアラートテンプレートは700です。</li><li>に適用できるアラートテンプレート監視対象サーバー上のデータベースの特定のスキーマのファンクションは800で |

表 604 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| param_names | [string] | false | none | パラメーターの名前を定義します。 |
| param_types | string | false | none | パラメーターのタイプを定義します。例 INTEGER/STRING.. |
| params_units | [string] | false | none | パラメーターに使用される単位を定義します。例 'DAYS'、'True or False'、'Hours'、'Minutes' など |
| sql | string | false | none | しきい値に到達したことを示す数値を取得するための SQL クエリー。パラメーター化される場合があります。注 このプロパティは、POST 要求の必須パラメーターです。 |
| applicable_on_server | string | false | none | テンプレートを適用する監視対象サーバーの種類を指定します。使用可能な値は、ALL、POSTGRES_SERVER、および ADVANCED_SERVER です。注 このプロパティは、POST 要求の必須パラメーターです。 |
| default_check_frequency | integer | false | none | このテンプレートで定義されたアラートをチェックする頻度を分単位で定義します。注 このプロパティは、POST 要求の必須パラメーターです。 |
| default_history_retention | integer | false | none | このタイプのアラートの履歴保持期間を日単位で定義します。注 このプロパティは、POST 要求の必須パラメーターです。 |
| probe_dependency_list | [string] | false | none | アラートテンプレートが依存するプローブのリストを定義します。 |

次のページに続く

表 604 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| threshold_unit | string | false | none | MB/kB/秒/日/... などの、しきい値の計算に使用される単位 この値は、GUI でのみ使用されます。 |
| is_system_template | boolean | false | none | 指定されたテンプレートがシステム定義であるかユーザー定義であるかを判断します。注このプロパティは、POST/PUT 要求には適用されません。 |
| is_auto_create | boolean | false | none | エージェントまたはサーバーが PEM に追加されたらすぐにアラートを作成するかどうかを決定します。注 エージェントおよびサーバーレベル、つまり object_type 100 および 200 でのみサポートされています。 |
| operator | string | false | none | しきい値を比較する演算子。エージェントおよびサーバーレベル、つまり object_type 100 および 200 でのみサポートされています。 |
| low_threshold_value | number | false | none | 低しきい値。エージェントおよびサーバーレベル、つまり object_type 100 および 200 でのみサポートされています。 |
| medium_threshold_value | number | false | none | 中しきい値。エージェントおよびサーバーレベル、つまり object_type 100 および 200 でのみサポートされています。 |
| high_threshold_value | number | false | none | 高しきい値。エージェントおよびサーバーレベル、つまり object_type 100 および 200 でのみサポートされています。 |

次のページに続く

表 604 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| info_sql | string | false | none | ダッシュボードのアラートに関する詳細情報を提供する SQL クエリーを指定します。 |

### 列挙値

| Property | Value |
|---|---|
| applicable_on_server | ALL |
| applicable_on_server | POSTGRES_SERVER |
| applicable_on_server | ADVANCED_SERVER |
| operator | > |
| operator | < |

### AlertTemplatePost


```
{
  "name": "string",
  "description": "string",
  "object_type": 0,
  "params": [
    {
      "param_types": "STRING",
      "param_units": "KK",
      "param_names": "param_1"
    }
  ],
  "sql": "string",
  "applicable_on_server": "ALL",
  "default_check_frequency": 0,
  "default_history_retention": 0,
  "probe_dependency_list": [
    {
      "internal_name": "probe_name"
    }
  ],
  "threshold_unit": "string",
  "is_system_template": false,
  "is_auto_create": true,
  "operator": ">",
  "low_threshold_value": 0,
  "medium_threshold_value": 0,
```


(次のページに続く)

(前のページからの続き)

```
  "high_threshold_value": 0,
  "info_sql": "string"
}
```

### プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| name | string | false | none | アラートテンプレートの名前。注 このプロパティは、POST 要求の必須パラメーターです。 |
| description | string | false | none | アラートテンプレートの説明。注 このプロパティは、POST 要求の必須パラメーターです。 |

次のページに続く

表 606 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| object_type | integer | false | none | このアラートテンプレートを適用するオブジェクトのタイプを定義します。<ul><li>システム全体のアラートテンプレートは50です。</li><li>エージェントに適用されるアラートテンプレートは100です。</li> <li>管理対象サーバーに適用されるアラートテンプレートは200です。</li><li>特定の監視対象サーバーのデータベースに適用されるアラートテンプレートは300です。</li><li>管理対象サーバーの特定のデータベースのスキーマに適用されるアラートテンプレートは400です。</li><li>特定のスキーマのテーブルに適用されるアラートテンプレート</li><li>監視対象サーバー上のデータベースの特定のスキーマのインデックスに適用されるアラートテンプレートは600です。>.</li><li>監視対象サーバー上のデータベースの特定のスキーマのシーケンスに適用できるアラートテンプレートは700です。</li><li>に適用できるアラートテンプレート監視対象サーバー上のデータベースの特定のスキーマのファンクションは800で |

表 606 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| params | [AlertTemplateParams] | false | none | アラートを確認するためにアラートテンプレートが使用するパラメーター/値のペアのリスト。注 アラートテンプレートには、all パラメーターを定義する必要があります。 |
| sql | string | false | none | しきい値に到達したことを示す数値を取得するための SQL クエリー。パラメーター化される場合があります。注 このプロパティは、POST 要求の必須パラメーターです。 |
| applicable_on_server | string | false | none | テンプレートを適用する監視対象サーバーの種類を指定します。使用可能な値は、ALL、POSTGRES_SERVER、および ADVANCED_SERVER です。注 このプロパティは、POST 要求の必須パラメーターです。 |
| default_check_frequency | integer | false | none | このテンプレートで定義されたアラートをチェックする頻度を分単位で定義します。注 このプロパティは、POST 要求の必須パラメーターです。 |
| default_history_retention | integer | false | none | このタイプのアラートの履歴保持期間を日単位で定義します。注 このプロパティは、POST 要求の必須パラメーターです。 |
| probe_dependency_list | [ProbeDependencyParams] | false | none | アラートテンプレートが依存するプローブのリストを定義します。 |

次のページに続く

表 606 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| threshold_unit | string | false | none | MB/kB/秒/日/... などの、しきい値の計算に使用される単位 この値は、GUI でのみ使用されます。 |
| is_system_template | boolean | false | none | 指定されたテンプレートがシステム定義であるかユーザー定義であるかを判断します。注このプロパティは、POST/PUT 要求には適用されません。 |
| is_auto_create | boolean | false | none | エージェントまたはサーバーが PEM に追加されたらすぐにアラートを作成するかどうかを決定します。注 エージェントおよびサーバーレベル、つまり object_type 100 および 200 でのみサポートされています。 |
| operator | string | false | none | しきい値を比較する演算子。エージェントおよびサーバーレベル、つまり object_type 100 および 200 でのみサポートされています。 |
| low_threshold_value | number | false | none | 低しきい値。エージェントおよびサーバーレベル、つまり object_type 100 および 200 でのみサポートされています。 |
| medium_threshold_value | number | false | none | 中しきい値。エージェントおよびサーバーレベル、つまり object_type 100 および 200 でのみサポートされています。 |
| high_threshold_value | number | false | none | 高しきい値。エージェントおよびサーバーレベル、つまり object_type 100 および 200 でのみサポートされています。 |

次のページに続く

表 606 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| info_sql | string | false | none | ダッシュボードのアラートに関する詳細情報を提供する SQL クエリーを指定します。 |

### 列挙値

| Property | Value |
|---|---|
| applicable_on_server | ALL |
| applicable_on_server | POSTGRES_SERVER |
| applicable_on_server | ADVANCED_SERVER |
| operator | > |
| operator | < |

### AlertTemplatePut

```
{
  "name": "string",
  "description": "string",
  "sql": "string",
  "applicable_on_server": "ALL",
  "default_check_frequency": 0,
  "default_history_retention": 0,
  "probe_dependency_list": {
    "deleted": [
      {
        "internal_name": "probe_name"
      }
    ],
    "added": [
      {
        "internal_name": "probe_name"
      }
    ]
  },
  "threshold_unit": "string",
  "is_system_template": false,
  "info_sql": "string"
}
```

プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| name | string | false | none | アラートテンプレートの名前。注 このプロパティは、POST 要求の必須パラメーターです。 |
| description | string | false | none | アラートテンプレートの説明。注 このプロパティは、POST 要求の必須パラメーターです。 |
| sql | string | false | none | しきい値に到達したことを示す数値を取得するための SQL クエリー。パラメーター化される場合があります。注 このプロパティは、POST 要求の必須パラメーターです。 |
| applicable_on_server | string | false | none | テンプレートを適用する監視対象サーバーの種類を指定します。使用可能な値は、ALL、POSTGRES_SERVER、および ADVANCED_SERVER です。注 このプロパティは、POST 要求の必須パラメーターです。 |
| default_check_frequency | integer | false | none | このテンプレートで定義されたアラートをチェックする頻度を分単位で定義します。注 このプロパティは、POST 要求の必須パラメーターです。 |
| default_history_retention | integer | false | none | このタイプのアラートの履歴保持期間を日単位で定義します。注 このプロパティは、POST 要求の必須パラメーターです。 |
| probe_dependency_list | object | false | none | アラートテンプレートが依存するプローブのリストを定義します。 |

次のページに続く

表 608 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » deleted | [ProbeDependencyParams] | false | none | 削除するパラメーター/値のペアのリスト。 |
| » added | [ProbeDependencyParams] | false | none | 追加するパラメーター/値のペアのリスト。 |
| threshold_unit | string | false | none | MB/kB/秒/日/... などの、しきい値の計算に使用される単位 この値は、GUI でのみ使用されます。 |
| is_system_template | boolean | false | none | 指定されたテンプレートがシステム定義であるかユーザー定義であるかを判断します。注このプロパティは、POST/PUT 要求には適用されません。 |
| info_sql | string | false | none | ダッシュボードのアラートに関する詳細情報を提供する SQL クエリーを指定します。 |

#### 列挙値

| Property | Value |
|---|---|
| applicable_on_server | ALL |
| applicable_on_server | POSTGRES_SERVER |
| applicable_on_server | ADVANCED_SERVER |

### AlertHistory

```
{
  "alert_id": 0,
  "state": "High",
  "value": "string",
  "actual_value": 0
}
```

### プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| alert_id | integer | false | none | アラート ID |
| state | string | false | none | 州 |
| value | string | false | none | ユーザーに提示できる価値 |
| actual_value | number | false | none | 状態変更時の実際の値 |

### 列挙値

| Property | Value |
|---|---|
| state | High |
| state | Medium |
| state | Low |

## ServerStatus


```
{
  "id": 0,
  "name": "string",
  "group_id": 0,
  "group_name": "string",
  "blackout": true,
  "status": "UP",
  "alerts": {
    "total": 0,
    "acknowledged": 0,
    "high": 0,
    "medium": 0,
    "low": 0,
    "high_acknowledged": 0,
    "medium_acknowledged": 0,
    "low_acknowledged": 0
  },
  "version": "string",
  "number_connections": 0,
  "remote_monitoring": true,
  "agent_id": 0,
  "sessions": {
    "last_recorded_time": 0,
    "sessions": [
      {
```


(次のページに続く)

(前のページからの続き)


```
        "database_name": "string",
        "procpid": 0,
        "usename": "string",
        "backend_start": 0,
        "xact_start": 0,
        "query_start": 0,
        "is_waiting": true,
        "is_idle": true,
        "is_idle_in_transaction": true,
        "is_vacuum": true,
        "is_autovacuum": true,
        "client_addr": "string",
        "client_port": 0,
        "memory_usage_mb": 0,
        "swap_usage_mb": 0,
        "cpu_usage": 0,
        "io_read_bytes": 0,
        "io_write_bytes": 0,
        "state": "string",
        "state_change": 0
      }
    ]
  }
}
```


### プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| id | integer | false | none | サーバー ID |
| name | string | false | none | サーバー名 |
| group_id | integer | false | none | データベースサーバーがユーザーに表示されるグループ ID |
| group_name | string | false | none | グループ名 |
| blackout | boolean | false | none | データベースサーバーのアラートブラックアウトのステータス |
| status | string | false | none | サーバーのステータスは UP、DOWN、UNKNOWN、または UNMANAGED です。 |

次のページに続く

表 612 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| alerts | object | false | none | データベースサーバーに構成されているアラートのリスト。アラートステータスと確認されたアラートの数を含む |
| » total | integer | false | none | データベースサーバーに構成されているアラートの数 |
| » acknowledged | integer | false | none | 確認されたアラートの数 |
| » high | integer | false | none | 状態が「高」のアラートの数 |
| » medium | integer | false | none | 状態が「Medium」のアラートの数 |
| » low | integer | false | none | 状態が「Low」のアラートの数 |
| » high_acknowledged | integer | false | none | 状態が「高」の確認されたアラートの数 |
| »medium_acknowledged | integer | false | none | 状態が「中」の確認されたアラートの数 |
| » low_acknowledged | integer | false | none | 状態が「Low」である確認されたアラートの数 |
| version | string | false | none | データベースサーバーの現在のバージョン |
| number_connections | integer | false | none | データベースサーバーの接続数 |
| remote_monitoring | boolean | false | none | データベースサーバーが pemAgent によってリモートで監視されているかどうかを判断します |
| agent_id | integer | false | none | データベースサーバーを監視しているエージェント ID |
| sessions | object | false | none | データベースサーバーのすべてのセッション情報 |
| » last_recorded_time | number | false | none | セッション情報の最後に記録された時間の UNIX 時間 |
| » sessions | [object] | false | none | データベースサーバーのすべてのセッション情報のリスト |
| » database_name | string | false | none | データベース名 |
| » procpid | integer | false | none | プロセス ID |
| » usename | string | false | none | ユーザー名 |

次のページに続く

表 612 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » backend_start | number | false | none | セッション開始時刻の UNIX 時間 |
| » xact_start | number | false | none | トランザクション開始時間の UNIX 時間 |
| » query_start | number | false | none | クエリ開始時間の UNIX 時間クエリが実行されている場合 |
| » is_waiting | boolean | false | none | セッションは待機中ですか？ |
| » is_idle | boolean | false | none | セッションはアイドルですか? |
| » is_idle_in_transaction | boolean | false | none | セッションはトランザクションでアイドル状態ですか? |
| » is_vacuum | boolean | false | none | これは VACUUM セッションですか？ |
| » is_autovacuum | boolean | false | none | これは AUTO VACUUM セッションですか? |
| » client_addr | string | false | none | セッションのクライアントのホストアドレス |
| » client_port | integer | false | none | セッションのクライアントポート |
| » memory_usage_mb | number | false | none | セッションのメモリ使用量 MB 単位ローカルに監視される場合にのみ使用可能 |
| » swap_usage_mb | number | false | none | セッションのスワップメモリ使用量 MB 単位ローカルに監視される場合にのみ使用可能 |
| » cpu_usage | number | false | none | セッションの CPU 使用率パーセンテージローカルに監視される場合にのみ使用可能 |
| » io_read_bytes | number | false | none | セッションの I/O 読み取りバイト単位ローカルに監視される場合にのみ使用可能 |
| » io_write_bytes | number | false | none | セッションの I/O 書き込みバイト単位ローカルに監視される場合にのみ使用可能 |

次のページに続く

表 612 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » state | string | false | none | セッションの現在の状態アイドル、アクティブ、トランザクション中のアイドルなど |
| » state_change | number | false | none | セッション内の最後の状態変更の UNIX 時間 |

#### 列挙値

| Property | Value |
|---|---|
| status | UP |
| status | DOWN |
| status | UNKNOWN |
| status | UNMANAGED |

### エージェントステータス

```
{
  "id": 0,
  "name": "string",
  "group_id": 0,
  "group_name": "string",
  "blackout": true,
  "status": "UP",
  "alerts": {
    "total": 0,
    "acknowledged": 0,
    "high": 0,
    "medium": 0,
    "low": 0,
    "high_acknowledged": 0,
    "medium_acknowledged": 0,
    "low_acknowledged": 0
  },
  "version": "string",
  "processes": 0,
  "threads": 0,
  "cpu_utilization": 0,
  "memory_utilization": 0,
  "swap_utilization": 0,
  "disk_utilization": 0
}
```

プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| id | integer | false | none | エージェント ID |
| name | string | false | none | エージェント名 |
| group_id | integer | false | none | エージェントがユーザーに表示されるグループ ID |
| group_name | string | false | none | グループ名 |
| blackout | boolean | false | none | エージェントのアラートブラックアウトを有効にするか無効にするかを決定します。 |
| status | string | false | none | エージェントのステータスは UP、DOWN、または UNKNOWN です |
| alerts | object | false | none | エージェントに構成されたアラートのリスト。アラートステータスと確認されたアラートの数を含む |
| » total | integer | false | none | エージェントに構成されたアラートの数 |
| » acknowledged | integer | false | none | 確認されたアラートの数 |
| » high | integer | false | none | 状態が「高」のアラートの数 |
| » medium | integer | false | none | 状態が「Medium」のアラートの数 |
| » low | integer | false | none | 状態が「Low」のアラートの数 |
| » high_acknowledged | integer | false | none | 状態が「高」の確認されたアラートの数 |
| »medium_acknowledged | integer | false | none | 状態が「中」の確認されたアラートの数 |
| » low_acknowledged | integer | false | none | 状態が「Low」である確認されたアラートの数 |
| version | string | false | none | エージェントの現在のバージョン |
| processes | integer | false | none | 実行中のプロセスの数 |
| threads | integer | false | none | 実行中のすべてのプロセスのスレッド数 |
| cpu_utilization | number | false | none | CPU 使用率パーセント |
| memory_utilization | number | false | none | メモリ使用率パーセント |
| swap_utilization | number | false | none | スワップメモリ使用率パーセント |

次のページに続く

表 614 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| disk_utilization | number | false | none | 全体のディスク使用率パーセント |

### 列挙値

| Property | Value |
|---|---|
| status | UP |
| status | DOWN |
| status | UNKNOWN |

### アラートステータス

```
{
  "alert_since": "string",
  "alert_id": 0,
  "alert_name": "string",
  "alert_state": "High",
  "value": "string",
  "object_description": "string",
  "server_id": 0,
  "agent_id": 0,
  "database": "string",
  "schema": "string",
  "package": "string",
  "object": "string",
  "alert_target_level": "Global",
  "last_processed": "string",
  "info": "string",
  "info_cols": ["string"],
  "info_vals": [["string"]]
}
```

### プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| alert_since | string | false | none | アラート状態が最後に変更された Unix 時間 |
| alert_id | integer | false | none | アラート内部識別番号 |
| alert_name | string | false | none | アラートの名前 |

次のページに続く

表 616 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| alert_state | string | false | none | アラートの現在の状態。'null' 値は、アラートのアラートしきい値が現在違反していないことを示します。 |
| value | string | false | none | アラートの現在値 |
| object_description | string | false | none | モニタリングオブジェクトの説明/名前 |
| server_id | integer | false | none | null またはサーバーの ID |
| agent_id | integer | false | none | null またはエージェントの ID |
| database | string | false | none | null またはデータベースの名前 |
| schema | string | false | none | null またはスキーマ/名前空間の名前 |
| package | string | false | none | null または EDB パッケージの名前 |
| object | string | false | none | null または他の種類のオブジェクトの名前 |
| alert_target_level | string | false | none | 警戒対象レベル |
| last_processed | string | false | none | アラートが処理された Unix 時間を表す文字列 |
| info | string | false | none | アラートが生成された時点の情報 |
| info_cols | [string] | false | none | 情報の列名の配列 |
| info_vals | [array] | false | none | テーブル形式のアラートに関連付けられた詳細情報配列の配列 |

### 列挙値

| Property | Value |
|---|---|
| alert_state | High |
| alert_state | Medium |
| alert_state | Low |
| alert_target_level | Global |
| alert_target_level | Agent |
| alert_target_level | Server |
| alert_target_level | Database |
| alert_target_level | Schema |
| alert_target_level | Table |

次のページに続く

表 617 – 前のページからの続き

| Property | Value |
|---|---|
| alert_target_level | Index |
| alert_target_level | Sequence |
| alert_target_level | Function |
| alert_target_level | View |

### CustomProbes


```
{
  "probe_id": 0,
  "probe_name": "string",
  "enabled": true,
  "target_type": 0,
  "internal_name": "string",
  "is_system_probe": true,
  "any_server_version": true,
  "collection_method": "w",
  "interval": 0,
  "lifetime": 0,
  "discard_history": true,
  "platform": "unix",
  "probe_code": "string",
  "probe_columns": [
    {
      "pc_id": 0,
      "pc_name": "string",
      "pc_data_type": "bigint",
      "pc_internal_name": "string",
      "pc_unit": "string",
      "pc_col_type": "m",
      "pc_position": 0,
      "pc_graphable": true,
      "pc_calc_pit": true,
      "pc_pit_default": true
    }
  ],
  "alternate_code": [
    {
      "server_version_id": 0,
      "server_probe_code": "string",
      "extension_version": 0
    }
  ]
}
```

プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| probe_id | integer | false | none | プローブの識別子 |
| probe_name | string | false | none | プローブの名前 |
| enabled | boolean | false | none | プローブがデフォルトで有効か無効かを決定します。 |

次のページに続く

表 618 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| target_type | integer | false | none | プローブが適用されるオブジェクトのタイプを定義します。<ul><li>target_type 100のプローブは、エージェントの統計を収集します。</li><li>target_type 200のプローブは、収集を担当します。</li><li>target_type 300のプローブは、データベースの統計を収集します。</li><li>target_type 400<のプローブ/b>は、スキーマの統計を収集します。</li><li>target_type 500のプローブは、テーブルの統計を収集します。</li><li>次のプローブtarget_type 600は、インデックスの統計を収集します。</li><li>target_type 700のプローブは、シーケンスの統計を収集します。</li> <li>target_type 800のプローブは、ファンクションの統計を収集します。</li><li>target_type 900のプローブは、ファンクションの統計を収集します。view.</li><li>target_type 1000のプローブは、エクステンションの統計を収集します。</li></ul> |

次のページに続く

表 618 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| internal_name | string | false | none | プローブの内部名を定義します。 PEM システムは、プローブ統計を'pem-data' および'pemhistory' スキーマに保存するための内部テーブルを作成します。これらのテーブルの名前は、内部名と同じです。 |
| is_system_probe | boolean | false | none | このプローブがユーザー定義であるかどうかを判断します。ユーザー定義プローブのこのフラグは常に false です。 |
| any_server_version | boolean | false | none | このプローブは Postgres バリアントのバージョンに適用されるかどうかを判断します。詳細については、ドキュメントを参照してください。 |

次のページに続く

表 618 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| collection_method | string | false | none | 統計収集の方法を定義します。 PEM は、カスタムメイドのプローブの 3 つの方法 SQL、WMI、またはバッチ/シェルスクリプトをサポートしています。<ul><li>'w' は WMI スクリプトを表します。このタイプのスクリプトは Windows プラットフォームにのみ適用され、target_type は 100エージェント用です。</li><li>'b' はバッチ/シェルスクリプトを表します。このスクリプトの'target_type' は常に 100エージェントです。このスクリプトは、 Windows ではバッチスクリプトとして、他のプラットフォームではシェルスクリプトとして動作します。</li><li>'s' は SQL スクリプトの略です。 'target_type' は常に 200サーバー、 300データベース、 400スキーマ、 500です。テーブル、 600インデックス、 700シーケンス、 800ファンクション、または 900 view)、または1000拡張機能。 </li></ul> |

次のページに続く

表 618 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| interval | integer | false | none | プローブが統計を収集する頻度を秒単位で定義します。10 以上である必要があります |
| lifetime | integer | false | none | 履歴データの保持期間を日単位で定義します。有効な範囲は 1〜365 です。 'discard_history' が true に設定されている場合、この値は効果がありません。 |
| discard_history | boolean | false | none | 履歴データを破棄するかどうかを決定します。'discard_history' が false に設定されている場合、このプローブのデータの履歴は保存されません。 |
| platform | string | false | none | このプローブを実行できるプラットフォームを定義します。この値は、エージェントレベルのプローブにのみ適用されます。 |
| probe_code | string | false | none | コードの定義に従って統計を収集するためのSQL/Batch スクリプト/シェルスクリプト/WMI スクリプト。 |
| probe_columns | [object] | false | none | なし |
| » pc_id | integer | false | none | 列識別子 |
| » pc_name | string | false | none | カラム名 |
| » pc_data_type | string | false | none | 列のデータ型。サポートされているデータ型は次のとおりです。<ul><ul>bigint</ul><ul>char</ul><ul>10 進 数</ul><ul>倍 精 度</ul><ul>整 数</ul><ul>数 値</ul><ul>実数</ul><ul>テ キ ス ト</ul><ul>text[]</ul><ul>timestamp</ul><ul>t/ul> |
| » pc_internal_name | string | false | none | 列の内部名を返します。 |
| » pc_unit | string | false | none | 列の単位を返します。 |

次のページに続く

表 618 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » pc_col_type | string | false | none | 列を内部テーブルの主キーの一部として返すか、否か。 'k' はキー列を表し、'm' はメンバー/ノーマル列を表します。 |
| » pc_position | integer | false | none | 内部テーブルの列の位置を返します。 |
| » pc_graphable | boolean | false | none | 列のデータをチャートにグラフ化できるかどうかを決定します。 |
| » pc_calc_pit | boolean | false | none | サーバーがこの列のメトリックデータのポイントインタイムを計算するかどうかを決定します。 |
| » pc_pit_default | boolean | false | none | メトリックをデフォルトでポイントインタイムで保存するかどうかを決定します。詳細については、ドキュメントの'Custom Probes' セクションを参照してください。 |
| alternate_code | [AlternateServerProbeCode ] | false | none | なし |

### 列挙値

| Property | Value |
|---|---|
| collection_method | w |
| collection_method | b |
| collection_method | s |
| platform | unix |
| platform | windows |
| pc_data_type | bigint |
| pc_data_type | char |
| pc_data_type | decimal |
| pc_data_type | double precision |
| pc_data_type | integer |
| pc_data_type | numeric |
| pc_data_type | real |
| pc_data_type | text |
| pc_data_type | text[] |

次のページに続く

表 619 – 前のページからの続き

| Property | Value |
|---|---|
| pc_data_type | timestamp |
| pc_data_type | timestamptz |
| pc_col_type | m |
| pc_col_type | k |

### CustomProbeExportData

```
{
  "probes": [0]
}
```

### プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| probes | [integer] | false | none | カスタムプローブのプローブ ID のリスト |

### CustomProbeImportData

```
{
  "version": 1,
  "skip_overwrite": true,
  "probes": [
    {
      "probe_name": "string",
      "collection_method": "w",
      "enabled": true,
      "target_type": 0,
      "any_server_version": true,
      "any_extension_version": true,
      "extension_name": "string",
      "interval": 0,
      "lifetime": 0,
      "discard_history": true,
      "platform": "unix",
      "probe_code": "string",
      "probe_columns": [
        {
          "pc_name": "string",
          "pc_data_type": "bigint",
          "pc_unit": "string",
```

(次のページに続く)

(前のページからの続き)


```
          "pc_col_type": "m",
          "pc_graphable": true,
          "pc_calc_pit": true,
          "pc_pit_default": true
        }
      ],
      "alternate_code": [
        {
          "server_version_id": 0,
          "server_probe_code": "string",
          "extension_version": 0
        }
      ],
      "internal_name": "string"
    }
  ]
}
```


### プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| version | integer | false | none | プローブエクスポートのバージョン |
| skip_overwrite | boolean | false | none | 既存のプローブをスキップできるかどうかを決定します。 |
| probes | [CustomProbeImportPostData ] | false | none | なし |

### CustomProbeImportPostData


```
{
  "probe_name": "string",
  "collection_method": "w",
  "enabled": true,
  "target_type": 0,
  "any_server_version": true,
  "any_extension_version": true,
  "extension_name": "string",
  "interval": 0,
  "lifetime": 0,
  "discard_history": true,
```


(次のページに続く)

(前のページからの続き)


```
  "platform": "unix",
  "probe_code": "string",
  "probe_columns": [
    {
      "pc_name": "string",
      "pc_data_type": "bigint",
      "pc_unit": "string",
      "pc_col_type": "m",
      "pc_graphable": true,
      "pc_calc_pit": true,
      "pc_pit_default": true
    }
  ],
  "alternate_code": [
    {
      "server_version_id": 0,
      "server_probe_code": "string",
      "extension_version": 0
    }
  ],
  "internal_name": "string"
}
```


### プロパティ

すべての

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| _anonymous_ | *CustomProbe-PostData* | false | none | なし |

そして

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| _anonymous_ | object | false | none | なし |
| » internal_name | string | false | none | プローブがエクスポートされたシステムからのプローブの内部名 |

### CustomProbePostData

```
{
  "probe_name": "string",
  "collection_method": "w",
  "enabled": true,
  "target_type": 0,
  "any_server_version": true,
  "any_extension_version": true,
  "extension_name": "string",
  "interval": 0,
  "lifetime": 0,
  "discard_history": true,
  "platform": "unix",
  "probe_code": "string",
  "probe_columns": [
    {
      "pc_name": "string",
      "pc_data_type": "bigint",
      "pc_unit": "string",
      "pc_col_type": "m",
      "pc_graphable": true,
      "pc_calc_pit": true,
      "pc_pit_default": true
    }
  ],
  "alternate_code": [
    {
      "server_version_id": 0,
      "server_probe_code": "string",
      "extension_version": 0
    }
  ]
}
```

プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| probe_name | string | false | none | プローブの名前 |

次のページに続く

表 624 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| collection_method | string | false | none | 統計収集の方法を定義します。 PEM は、カスタムメイドのプローブの 3 つの方法 SQL、WMI、またはバッチ/シェルスクリプトをサポートしています。<ul><li>'w' は WMI スクリプトを表します。このタイプのスクリプトは Windows プラットフォームにのみ適用され、target_type は 100エージェント用です。</li><li>'b' はバッチ/シェルスクリプトを表します。このスクリプトの'target_type' は常に 100エージェントです。このスクリプトは、 Windows ではバッチスクリプトとして、他のプラットフォームではシェルスクリプトとして動作します。</li><li>'s' は SQL スクリプトの略です。 'target_type' は常に 200サーバー、 300データベース、 400スキーマ、 500です。テーブル、 600 インデックス、 700 シーケンス、 800ファンクション、 900 ビュー) または1000拡張子。</li></ul> |

次のページに続く

表 624 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| enabled | boolean | false | none | プローブがデフォルトで有効/無効かどうかを決定します。 |

次のページに続く

表 624 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| target_type | integer | false | none | プローブが適用されるオブジェクトのタイプを定義します。<ul><li>target_type 100のプローブは、エージェントの統計を収集します。</li><li>target_type 200のプローブは、収集を担当します。</li><li>target_type 300のプローブは、データベースの統計を収集します。</li><li>target_type 400<のプローブ/b>は、スキーマの統計を収集します。</li><li>target_type 500のプローブは、テーブルの統計を収集します。</li><li>次のプローブtarget_type 600は、インデックスの統計を収集します。</li><li>target_type 700のプローブは、シーケンスの統計を収集します。</li> <li>target_type 800のプローブは、ファンクションの統計を収集します。</li><li>target_type 900のプローブは、ファンクションの統計を収集します。view.</li><li>target_type 1000のプローブは、エクステンの統計を収集します。</li></ul> |

次のページに続く

表 624 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| any_server_version | boolean | false | none | このプローブは Postgres バリアントのバージョンに適用されるかどうかを判断します。詳細については、ドキュメントを参照してください。 |
| any_extension_version | boolean | false | none | このプローブは拡張バリアントのバージョンに適用されるかどうかを判断します。詳細については、ドキュメントを参照してください。 |
| extension_name | string | false | none | 拡張機能の名前。<ul>有効な名前は次のとおりです。 edbspl 、 plpgsql 、 hstore 、 pldbgapi 、 edb_dblink_oci 、 edb_dblink_libpq 、 sslutils .</ul> |
| interval | integer | false | none | プローブが統計を収集する頻度を秒単位で定義します。10 以上である必要があります |
| lifetime | integer | false | none | 履歴データの保持期間を日単位で定義します。有効な範囲は 1〜365 です。 'discard_history' が true に設定されている場合、この値は効果がありません。 |
| discard_history | boolean | false | none | 履歴データを破棄するかどうかを定義します。 'discard_history' が false に設定されている場合、このプローブの履歴データは保存されません。 |

次のページに続く

表 624 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| platform | string | false | none | このプローブを実行できるプラットフォームを定義します。この値は、エージェントレベルのプローブにのみ適用されます。 |
| probe_code | string | false | none | コードの定義ごとに統計を収集するためのSQL/Batch スクリプト/シェルスクリプト/WMI スクリプト。 |
| probe_columns | [object] | false | none | なし |
| » pc_name | string | false | none | カラム名 |
| » pc_data_type | string | false | none | 列のデータ型。サポートされているデータ型は次のとおりです。<ul><ul>bigint</ul><ul>char</ul><ul>10 進 数</ul><ul>倍 精 度</ul><ul>整 数</ul><ul>数 値</ul><ul>実数</ul><ul>テ キ ス ト</ul><ul>text[]</ul><ul>timestamp</ul><ul>t/ul> |
| » pc_unit | string | false | none | 列の単位を返します。 |
| » pc_col_type | string | false | none | 内部テーブルの主キーの一部として列を返すか、否か。 'k' はキー列を表し、'm' はメンバー/ノーマル列を表します。 |
| » pc_graphable | boolean | false | none | 列のデータをチャートにグラフ化できるかどうかを決定します。 |
| » pc_calc_pit | boolean | false | none | サーバーがこの列のメトリックデータのポイントインタイムを計算するかどうかを決定します。 |

次のページに続く

表 624 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » pc_pit_default | boolean | false | none | メトリックデータをデフォルトでポイントインタイムで保存するかどうかを決定します。詳細については、ドキュメントの'Custom Probes'セクションを参照してください。 |
| alternate_code | [Alternate-ServerProbe-Code ] | false | none | なし |

### 列挙値

| Property | Value |
|---|---|
| collection_method | w |
| collection_method | b |
| collection_method | s |
| platform | unix |
| platform | windows |
| pc_data_type | bigint |
| pc_data_type | char |
| pc_data_type | decimal |
| pc_data_type | double precision |
| pc_data_type | integer |
| pc_data_type | numeric |
| pc_data_type | real |
| pc_data_type | text |
| pc_data_type | text[] |
| pc_data_type | timestamp |
| pc_data_type | timestamptz |
| pc_col_type | m |
| pc_col_type | k |

### CustomProbePutData

```
{
  "interval_min": 0,
  "interval_sec": 0,
  "lifetime": 0,
  "enabled": true,
  "probe_code": "string",
```

(次のページに続く)

(前のページからの続き)


```
  "probe_columns": {
    "changed": [
      {
        "pc_id": 0,
        "pc_unit": "string",
        "pc_graphable": true
      }
    ]
  },
  "alternate_code": {
    "changed": [
      {
        "server_version_id": 0,
        "server_probe_code": "string",
        "extension_version": 0
      }
    ],
    "deleted": [
      {
        "server_version_id": 0
      }
    ],
    "added": [
      {
        "server_version_id": 0,
        "server_probe_code": "string",
        "extension_version": 0
      }
    ]
  }
}
```


### プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| interval_min | integer | false | none | プローブが統計を収集する頻度を分単位で定義します。指定できる値は 0 ～2880 です。 |
| interval_sec | integer | false | none | プローブが統計を収集する頻度を秒単位で定義します。指定できる値は 0 ～59 です。 |

次のページに続く

表 626 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| lifetime | integer | false | none | 履歴データの保持期間を日単位で定義します。有効な範囲は 1〜365 です。 'discard_history' が true に設定されている場合、この値は効果がありません。 |
| enabled | boolean | false | none | プローブがデフォルトで有効になっているかどうかを決定します。 |
| probe_code | string | false | none | コードの定義ごとに統計を収集するための SQL/Batch スクリプト/シェルスクリプト/WMI スクリプト。 |
| probe_columns | object | false | none | なし |
| » changed | [ProbeColumn ] | false | none | 変更されたプローブカラムのリスト |
| alternate_code | object | false | none | なし |
| » changed | [Alternate-ServerProbe-Code ] | false | none | 一部のサーバーバージョンの変更された代替プローブコードのリスト |
| » deleted | [DeletedAl-ternateServer-ProbeCode ] | false | none | 一部のサーバーバージョンの削除された代替プローブコードのリスト |
| » added | [Alternate-ServerProbe-Code ] | false | none | 一部のサーバーバージョンで追加された代替プローブコードのリスト |

### プローブ

```
{
  "probe_id": 0,
  "probe_name": "string",
  "default_interval_min": 0,
  "interval_min": 0,
  "default_interval_sec": 0,
  "interval_sec": 0,
  "default_interval": 0,
  "interval": 0,
  "default_lifetime": 0,
  "lifetime": 0,
```

(次のページに続く)

(前のページからの続き)

```
  "default_enabled": true,
  "enabled": true,
  "target_type_id_returned": 0
}
```

### プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| probe_id | integer | false | none | プローブの一意の ID |
| probe_name | string | false | none | プローブの名前 |
| default_interval_min | integer | false | none | プローブが統計を収集するデフォルトの間隔を分単位で定義します。指定できる値は 0〜2880 です。 |
| interval_min | integer | false | none | プローブが統計を収集する頻度を分単位で定義します。指定できる値は 0〜2880 です。 |
| default_interval_sec | integer | false | none | プローブが統計を収集するデフォルトの間隔を秒単位で定義します。指定できる値は 0〜59 です。 |
| interval_sec | integer | false | none | プローブが統計を収集する頻度を秒単位で定義します。指定できる値は 0〜59 です。 |
| default_interval | integer | false | none | プローブが統計を収集するデフォルトの間隔を秒単位で定義します。 |
| interval | integer | false | none | プローブが統計を収集する頻度を秒単位で定義します。10 以上である必要があります |
| default_lifetime | integer | false | none | デフォルトの履歴データの保持期間を日単位で定義します。 |

次のページに続く

表 627 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| lifetime | integer | false | none | 履歴データの保持期間を日単位で定義します。有効な範囲は 1〜365 です。 'discard_history' が true に設定されている場合、この値は効果がありません。 |
| default_enabled | boolean | false | none | デフォルトのプローブ有効/無効値を定義します。 |
| enabled | boolean | false | none | プローブがデフォルトで有効/無効かどうかを指定します。 |

次のページに続く

表 627 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| target_type_id_returned | integer | false | none | 戻りターゲットタイプの値を定義します。<ul><li>target_type 100のプローブは、エージェントの統計を収集します。</li><li>target_type 200のプローブは、収集を担当します。</li><li>target_type 300のプローブは、データベースの統計を収集します。</li><li>target_type 400<のプローブ/b>は、スキーマの統計を収集します。</li><li>target_type 500のプローブは、テーブルの統計を収集します。</li><li>次のプローブtarget_type 600は、インデックスの統計を収集します。</li><li>target_type 700のプローブは、シーケンスの統計を収集します。</li> <li>target_type 800のプローブは、ファンクションの統計を収集します。</li><li>target_type 900のプローブは、ファンクションの統計を収集します。view.</li></ul> |

### ProbeConfigData


```
{
  "interval_min": 0,
  "interval_sec": 0,
  "lifetime": 0,
  "enabled": true
}
```


#### プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| interval_min | integer | false | none | プローブの実行頻度分単位。許可される間隔値分単位は 0〜2880 です。注合計間隔は 10 秒より大きい必要があります。つまり、 10 >= interval_min * 60 + interval_sec >= 172800 |
| interval_sec | integer | false | none | プローブの実行頻度秒単位。許可される間隔値秒単位は 0〜59 です。注合計間隔は 10 秒より大きい必要があります。つまり、 10 >= interval_min * 60 + interval_sec >= 172800 |
| lifetime | integer | false | none | プローブ履歴データの保持期間日単位。つまり、1 <= ライフタイム <= 365 |
| enabled | boolean | false | none | 指定されたオブジェクトでプローブが有効になっているかどうかを判断します。 |

### CopyAgentData


```
{
  "type": "string",
  "agent_id": 0
}
```

プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| type | string | false | none | 値は「agent」である必要があります |
| agent_id | integer | false | none | なし |

### CopyServerData

```
{
  "type": "string",
  "server_id": 0
}
```

プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| type | string | false | none | 値は「server」である必要があります |
| server_id | integer | false | none | なし |

### CopyDatabaseData

```
{
  "type": "string",
  "server_id": 0,
  "database_name": "string"
}
```

プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| type | string | false | none | 値は「database」である必要があります |
| server_id | integer | false | none | なし |
| database_name | string | false | none | なし |

### CopySchemaData

```
{
  "type": "string",
  "server_id": 0,
  "database_name": "string",
  "schema_name": "string"
}
```

#### プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| type | string | false | none | 値は「スキーマ」である必要があります |
| server_id | integer | false | none | なし |
| database_name | string | false | none | なし |
| schema_name | string | false | none | なし |

### CopyAlertAgentData

```
{
  "type": "string",
  "agent_id": 0,
  "existing_alert_options": "I"
}
```

#### プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| type | string | false | none | 値は「agent」である必要があります |
| agent_id | integer | false | none | なし |

次のページに続く

表 633 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| existing_alert_options | string | false | none | コピーする重複したアラートを無視/置換するか、既存のすべてのアラートをターゲットから削除するかどうかを決定します。サポートされている有効な値は次のとおりです。I -> Ignore Duplicates R -> Replace Duplicates D -> Delete Existing Alerts |

### 列挙値

| Property | Value |
|---|---|
| existing_alert_options | I |
| existing_alert_options | R |
| existing_alert_options | D |

### CopyAlertServerData

```
{
  "type": "string",
  "server_id": 0,
  "existing_alert_options": "I"
}
```

### プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| type | string | false | none | 値は「server」である必要があります |
| server_id | integer | false | none | なし |

次のページに続く

表 635 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| existing_alert_options | string | false | none | コピーする重複したアラートを無視/置換するか、既存のすべてのアラートをターゲットから削除するかどうかを決定します。サポートされている有効な値は次のとおりです。I -> Ignore Duplicates R -> Replace Duplicates D -> Delete Existing Alerts |

#### 列挙値

| Property | Value |
|---|---|
| existing_alert_options | I |
| existing_alert_options | R |
| existing_alert_options | D |

### CopyAlertDatabaseData

```
{
  "type": "string",
  "server_id": 0,
  "database_name": "string",
  "existing_alert_options": "I"
}
```

#### プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| type | string | false | none | 値は「database」である必要があります |
| server_id | integer | false | none | なし |
| database_name | string | false | none | なし |

次のページに続く

表 637 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| existing_alert_options | string | false | none | コピーする重複したアラートを無視/置換するか、既存のすべてのアラートをターゲットから削除するかどうかを決定します。サポートされている有効な値は次のとおりです。I -> Ignore Duplicates R -> Replace Duplicates D -> Delete Existing Alerts |

### 列挙値

| Property | Value |
|---|---|
| existing_alert_options | I |
| existing_alert_options | R |
| existing_alert_options | D |

### CopyAlertSchemaData

```
{
  "type": "string",
  "server_id": 0,
  "database_name": "string",
  "schema_name": "string",
  "existing_alert_options": "I"
}
```

### プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| type | string | false | none | 値は「スキーマ」である必要があります |
| server_id | integer | false | none | なし |
| database_name | string | false | none | なし |
| schema_name | string | false | none | なし |

次のページに続く

表 639 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| existing_alert_options | string | false | none | コピーする重複したアラートを無視/置換するか、既存のすべてのアラートをターゲットから削除するかどうかを決定します。サポートされている有効な値は次のとおりです。I -> Ignore Duplicates R -> Replace Duplicates D -> Delete Existing Alerts |

#### 列挙値

| Property | Value |
|---|---|
| existing_alert_options | I |
| existing_alert_options | R |
| existing_alert_options | D |

### CopyAlertTableData

```
{
  "type": "string",
  "server_id": 0,
  "database_name": "string",
  "schema_name": "string",
  "table_name": "string",
  "existing_alert_options": "I"
}
```

#### プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| type | string | false | none | 値は「table」である必要があります |
| server_id | integer | false | none | なし |
| database_name | string | false | none | なし |
| schema_name | string | false | none | なし |
| table_name | string | false | none | なし |

次のページに続く

表 641 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| existing_alert_options | string | false | none | コピーする重複したアラートを無視/置換するか、既存のすべてのアラートをターゲットから削除するかどうかを決定します。サポートされている有効な値は次のとおりです。I -> Ignore Duplicates R -> Replace Duplicates D -> Delete Existing Alerts |

### 列挙値

| Property | Value |
|---|---|
| existing_alert_options | I |
| existing_alert_options | R |
| existing_alert_options | D |

### CopyAlertIndexData

```
{
  "type": "string",
  "server_id": 0,
  "database_name": "string",
  "schema_name": "string",
  "index_name": "string",
  "existing_alert_options": "I"
}
```

### プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| type | string | false | none | 値は"index"である必要があります |
| server_id | integer | false | none | なし |
| database_name | string | false | none | なし |
| schema_name | string | false | none | なし |
| index_name | string | false | none | なし |

次のページに続く

表 643 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| existing_alert_options | string | false | none | コピーする重複したアラートを無視/置換するか、既存のすべてのアラートをターゲットから削除するかどうかを決定します。サポートされている有効な値は次のとおりです。I -> Ignore Duplicates R -> Replace Duplicates D -> Delete Existing Alerts |

#### 列挙値

| Property | Value |
|---|---|
| existing_alert_options | I |
| existing_alert_options | R |
| existing_alert_options | D |

### CopyAlertFunctionData

```
{
  "type": "string",
  "server_id": 0,
  "database_name": "string",
  "schema_name": "string",
  "function_name": "string",
  "function_arguments": "string",
  "existing_alert_options": "I"
}
```

#### プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| type | string | false | none | 値は「ファンクション」である必要があります |
| server_id | integer | false | none | なし |
| database_name | string | false | none | なし |
| schema_name | string | false | none | なし |
| function_name | string | false | none | なし |

次のページに続く

表 645 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| function_arguments | string | false | none | なし |
| existing_alert_options | string | false | none | コピーする重複したアラートを無視/置換するか、既存のすべてのアラートをターゲットから削除するかどうかを決定します。サポートされている有効な値は次のとおりです。I -> Ignore Duplicates R -> Replace Duplicates D -> Delete Existing Alerts |

#### 列挙値

| Property | Value |
|---|---|
| existing_alert_options | I |
| existing_alert_options | R |
| existing_alert_options | D |

### CopyAlertSequenceData

```
{
  "type": "string",
  "server_id": 0,
  "database_name": "string",
  "schema_name": "string",
  "sequence_name": "string",
  "existing_alert_options": "I"
}
```

#### プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| type | string | false | none | 値は"sequence"である必要があります |
| server_id | integer | false | none | なし |
| database_name | string | false | none | なし |
| schema_name | string | false | none | なし |
| sequence_name | string | false | none | なし |

次のページに続く

表 647 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| existing_alert_options | string | false | none | コピーする重複したアラートを無視/置換するか、既存のすべてのアラートをターゲットから削除するかどうかを決定します。サポートされている有効な値は次のとおりです。I -> Ignore Duplicates R -> Replace Duplicates D -> Delete Existing Alerts |

### 列挙値

| Property | Value |
|---|---|
| existing_alert_options | I |
| existing_alert_options | R |
| existing_alert_options | D |

### エージェント

```
{
  "id": 0,
  "name": "string",
  "version": "string",
  "profile_id": 0,
  "active": true,
  "heartbeat_tolerance": 0,
  "alert_blackout": true,
  "owner": "string",
  "team": "string",
  "gid": 0,
  "tags": ["string"],
  "ignore_mnt_points": ["string"]
}
```

エージェント情報、バージョン、機能、およびエージェントがインストールされているホストに関する詳細を含むオブジェクトのリストを返します。

**例** \

```
[{
```

(次のページに続く)

(前のページからの続き)


```
  /* Agent information */
  "id": 1,
  "gid": 0,
  "name": "Postgres Enterprise Manager Host"
  "version": "9.3.0",
  "active": True,
  "heartbeat_tolerance": 15,
  "alert_blackout": False,
  "owner": "postgres",
  "team": None,

  /* Deatils about the Host, on which it is installed */
  "operating_sys": "Ubuntu 12.04.5 LTS",
  "host_name": "edb-ubuntu",
  "boot_time": "2016-09-19 12:28:27+05:30",
   "domain_name": "(none)",
   "window_domain": None,

  /* Agent capabilities */
  "capability_list": "unix, getloadavg, disk_space, memory_usage, cpu_usage, network_
↪statistics, io_analysis, data_log_file_analysis, disk_busy_info,
os_statistics, os_info, slony_replication, auto_discover_servers, allow_server_restart
↪",

  /* Monitored servers */
  "server_bind": "Postgres Enterprise Manager Server, PostgreSQL 11, PostgreSQL 12"
}]
```


ア

### プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| id | integer | false | none | pem エージェントの ID |
| name | string | false | none | エージェントの説明名前。 |
| version | string | false | none | pem-agent のバージョン |
| profile_id | integer | false | none | このエージェントを指定されたプロファイル ID で管理する必要があるかどうかを判断します。 |

次のページに続く

表 649 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| active | boolean | false | none | エージェントがアクティブかどうかを判断します。アクティブエージェントは、ホストおよび監視対象サーバーの統計を収集します。 |
| heartbeat_tolerance | integer | false | none | エージェントのハートビート許容秒単位。ハートビートがどれだけ遅れた場合、逃したと見なされるかを指定します。 |
| alert_blackout | boolean | false | none | エージェントのアラートをブラックアウトするかどうかを判断します。 |
| owner | string | false | none | エージェントをインストールしたユーザーのユーザー名。このユーザーを所有者と呼びます。 |
| team | string | false | none | このエージェントを表示できるロール/ユーザーの名前。 null/空の値は、すべてのユーザーがこのエージェントを表示できることを示します。 |
| gid | integer | false | none | グループの ID |
| tags | [string] | false | none | エージェントに関連付けられたタグのリスト |
| ignore_mnt_points | [string] | false | none | ディスク使用量の計算でマウントポイントを無視するために使用されます |

### エージェント


```
[
  {
    "id": 0,
    "name": "string",
    "version": "string",
    "profile_id": 0,
    "active": true,
    "heartbeat_tolerance": 0,
```


(次のページに続く)

(前のページからの続き)

```
    "alert_blackout": true,
    "owner": "string",
    "team": "string",
    "gid": 0,
    "tags": ["string"],
    "ignore_mnt_points": ["string"]
  }
]
```

エージェント情報、バージョン、機能、およびエージェントがインストールされているホストに関する詳細を含むオブジェクトのリストを返します。

例\

```
[{
  / *Agent information* /
  "id": 1,
  "gid": 0,
  "name": "Postgres Enterprise Manager Host"
  "version": "9.3.0",
  "active": True,
  "heartbeat_tolerance": 15,
  "alert_blackout": False,
  "owner": "postgres",
  "team": None,

  / *Deatils about the Host, on which it is installed* /
  "operating_sys": "Ubuntu 12.04.5 LTS",
  "host_name": "edb-ubuntu",
  "boot_time": "2016-09-19 12:28:27+05:30",
   "domain_name": "(none)",
   "window_domain": None,

  / *Agent capabilities* /
  "capability_list": "unix, getloadavg, disk_space, memory_usage, cpu_usage, network_
↪statistics, io_analysis, data_log_file_analysis, disk_busy_info,
os_statistics, os_info, slony_replication, auto_discover_servers, allow_server_restart
↪",

  / *Monitored servers* /
  "server_bind": "Postgres Enterprise Manager Server, PostgreSQL 11, PostgreSQL 12"
}]
```

ア

プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| id | integer | true | none | pem エージェントの ID |
| name | string | true | none | エージェントの説明名前。 |
| version | string | true | none | pem-agent のバージョン |
| profile_id | integer | false | none | このエージェントを指定されたプロファイル ID で管理する必要があるかどうかを判断します。 |
| active | boolean | true | none | エージェントがアクティブかどうかを判断します。アクティブエージェントは、ホストおよび監視対象サーバーの統計を収集します。 |
| heartbeat_tolerance | integer | true | none | エージェントのハートビート許容秒単位。ハートビートがどれだけ遅れた場合、逃したと見なされるかを指定します。 |
| alert_blackout | boolean | true | none | エージェントのアラートをブラックアウトするかどうかを判断します。 |
| owner | string | true | none | エージェントをインストールしたユーザーのユーザー名。このユーザーを所有者と呼びます。 |
| team | string | true | none | このエージェントを表示できるロール/ユーザーの名前。 null/空の値は、すべてのユーザーがこのエージェントを表示できることを示します。 |
| gid | integer | false | none | グループの ID |
| tags | [string] | false | none | エージェントに関連付けられたタグのリスト |
| ignore_mnt_points | [string] | false | none | ディスク使用量の計算でマウントポイントを無視するために使用されます |

サーバー


```
{
  "id": 0,
  "name": "string",
  "host": "string",
  "port": 0,
  "db": "string",
  "profile_id": 0,
  "serviceid": "string",
  "connection_params": [
    {
      "name": "string",
      "keyword": "string",
      "value": null
    }
  ],
  "active": true,
  "alert_blackout": true,
  "owner": 0,
  "team": "string",
  "server_owner": "string",
  "is_remote_monitoring": true,
  "tags": ["string"],
  "replication_solution": "none",
  "efm_cluster_name": "string",
  "efm_service_name": "string",
  "efm_installation_path": "string",
  "patroni_cluster_name": "string",
  "patroni_installation_path": "string",
  "patroni_config_path": "string",
  "comment": "string",
  "username": "string",
  "gid": 0,
  "server_group_name": "string",
  "db_res": "string",
  "role": "string",
  "is_edb": true,
  "agent_id": 0,
  "asb_host": "string",
  "asb_port": 0,
  "asb_username": "string",
  "asb_database": "string",
  "asb_sslmode": "prefer",
  "agent_allowtakeover": false,
  "agent_capability_list": ["string"],
```


(次のページに続く)

(前のページからの続き)


```
  "agent_description": "string",
  "sslcompression": false,
  "use_ssh_tunnel": false,
  "tunnel_host": "string",
  "tunnel_port": 22,
  "tunnel_username": "string",
  "tunnel_authentication": false,
  "tunnel_identity_file": "string",
  "tunnel_keep_alive": 0,
  "post_connection_sql": "string"
}
```


### プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| id | integer | false | none | サーバー ID. 注 このプロパティは、POST 要求には適用されません。 |
| name | string | false | none | サーバー名 |
| host | string | false | none | サーバーホスト |
| port | integer | false | none | サーバーポート |
| db | string | false | none | メンテナンスデータベース |
| profile_id | integer | false | none | プロファイル ID |
| serviceid | string | false | none | サーバーサービス ID |
| connection_params | [object] | false | none | サーバーの接続パラメーターのリスト。 |
| » name | string | false | none | パラメーター名 |
| » keyword | string | false | none | パラメーターキーワード |
| » value | any | false | none | パラメーター値 |
| active | boolean | false | none | サーバーがアクティブかどうかを判断します。 |
| alert_blackout | boolean | false | none | サーバー上のアラートがブラックアウトされているかどうかを判断します。 |
| owner | integer | false | none | サーバーを PEM に登録した所有者の ID。 |
| team | string | false | none | サーバー情報を表示できるロール/ユーザー名を定義します。 |

次のページに続く

表 651 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| server_owner | string | false | none | PEM データベースサーバーにサーバーを登録したユーザーの名前。 |
| is_remote_monitoring | boolean | false | none | このサーバーがリモートで監視されているかどうかを判断します。つまり、監視 pem エージェントが監視対象サーバーと同じホストにありません。 |
| tags | [string] | false | none | サーバーに関連付けられたタグのリスト。 |
| replication_solution | string | false | none | サーバーが使用するレプリケーションソリューション |
| efm_cluster_name | string | false | none | EFM のクラスター名 |
| efm_service_name | string | false | none | EFM のサービス名 |
| efm_installation_path | string | false | none | EFM のインストールパス |
| patroni_cluster_name | string | false | none | Patroni のクラスター名 |
| patroni_installation_path | string | false | none | Patroni のインストールパス |
| patroni_config_path | string | false | none | Patroni の構成パス |
| comment | string | false | none | サーバーの説明 |
| username | string | false | none | データベースサーバーに接続するためのユーザー名 |
| gid | integer | false | none | サーバーグループの ID |
| server_group_name | string | false | none | サーバーグループの名前 |
| db_res | string | false | none | データベース制限 SQL 句 |
| role | string | false | none | 接続が使用するロール |
| is_edb | boolean | false | none | サーバーは EDB Advanced Server ですか？ |
| agent_id | integer | false | none | サーバーにバインドされたエージェント ID |
| asb_host | string | false | none | PEM 接続の作成中にエージェントが使用する PEM データベースホスト |

次のページに続く

表 651 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| asb_port | integer | false | none | PEM 接続の作成中にエージェントが使用する PEM データベースポート |
| asb_username | string | false | none | PEM 接続の作成中にエージェントが使用する PEM データベースのユーザー名 |
| asb_database | string | false | none | エージェントが接続する PEM データベース |
| asb_sslmode | string | false | none | PEM データベース SSL モード |
| agent_allowtakeover | boolean | false | none | サーバーを別のエージェントに引き継ぎできるかどうかを指定します。 |
| agent_capability_list | [string] | false | none | エージェント機能の配列 |
| agent_description | string | false | none | エージェントの説明 |
| sslcompression | boolean | false | none | SSL 経由で送信されるデータを圧縮するかどうかを決定します。 |
| use_ssh_tunnel | boolean | false | none | ssh トンネルを使用するかどうかを決定します。 |
| tunnel_host | string | false | none | トンネルホスト名 |
| tunnel_port | integer | false | none | トンネルポート番号 |
| tunnel_username | string | false | none | トンネルのユーザー名 |
| tunnel_authentication | boolean | false | none | トンネル認証を使用するかどうかを決定します。 |
| tunnel_identity_file | string | false | none | 認証に使用されるトンネル ID ファイル。 |
| tunnel_keep_alive | integer | false | none | ssh トンネルのキープアライブパケットを送信する秒単位の間隔。 |
| post_connection_sql | string | false | none | 接続の確立後に実行される SQL 。 |

#### 列挙値

| Property | Value |
|---|---|
| replication_solution | none |
| replication_solution | efm |
| replication_solution | patroni |
| asb_sslmode | prefer |

次のページに続く

表 652 – 前のページからの続き

| Property | Value |
|---|---|
| asb_sslmode | allow |
| asb_sslmode | disable |
| asb_sslmode | require |
| asb_sslmode | verify-ca |
| asb_sslmode | verify-full |

### ServerGroupBody

```
{
  "name": "string"
}
```

#### プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| name | string | false | none | サーバーグループ名 |

### ServerGroupBodyGet

```
{
  "id": 0,
  "name": "string",
  "hidden": true,
  "parent_id": 0
}
```

#### プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| id | integer | false | none | サーバーグループ ID |
| name | string | false | none | サーバーグループ名 |
| hidden | boolean | false | none | Server グループを非表示にするかどうかを判断します |
| parent_id | integer | false | none | 親サーバーグループ ID。注 クラスタータイプのノードは親 ID を持ち、それ以外の場合は null になります。 |

### ServerBody


```
{
  "name": "string",
  "host": "string",
  "port": 0,
  "db": "string",
  "profile_id": 0,
  "serviceid": "string",
  "connection_params": [
    {
      "name": "sslmode",
      "keyword": "sslmode",
      "value": "prefer"
    },
    {
      "name": "connect_timeout",
      "keyword": "connect_timeout",
      "value": 10
    }
  ],
  "alert_blackout": true,
  "team": "string",
  "is_remote_monitoring": true,
  "tags": [
    {
      "text": "tag1",
      "color": "#8789AC"
    },
    {
      "text": "tag2",
      "color": "#0C767A"
    }
  ],
  "replication_solution": "none",
  "efm_cluster_name": "string",
  "efm_service_name": "string",
  "efm_installation_path": "string",
  "patroni_cluster_name": "string",
  "patroni_installation_path": "string",
  "patroni_config_path": "string",
  "comment": "string",
  "username": "string",
  "gid": 0,
  "hostaddr": "string",
  "db_res": "string",
```


(次のページに続く)

(前のページからの続き)


```
  "role": "string",
  "agent_id": 0,
  "asb_host": "string",
  "asb_port": 0,
  "asb_username": "string",
  "asb_database": "string",
  "asb_sslmode": "prefer",
  "asb_password": "string",
  "agent_allowtakeover": false,
  "fgcolor": "#000000",
  "bgcolor": "#FFFFFF",
  "sslcompression": false,
  "use_ssh_tunnel": false,
  "tunnel_host": "string",
  "tunnel_port": 22,
  "tunnel_username": "string",
  "tunnel_password": "string",
  "tunnel_authentication": false,
  "tunnel_identity_file": "string",
  "tunnel_keep_alive": 0,
  "post_connection_sql": "string"
}
```


### プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| name | string | false | none | サーバー名 |
| host | string | false | none | サーバーホスト名またはアドレス |
| port | integer | false | none | サーバーポート |
| db | string | false | none | メンテナンスデータベース |
| profile_id | integer | false | none | プロファイル ID |
| serviceid | string | false | none | サーバーサービス ID |
| connection_params | [object] | false | none | サーバーの接続パラメーターのリスト。 |
| » name | string | false | none | パラメーター名 |
| » keyword | string | false | none | パラメーターキーワード |
| » value | any | false | none | パラメーター値 |
| alert_blackout | boolean | false | none | サーバー上のアラートがブラックアウトされているかどうかを判断します。 |

次のページに続く

表 655 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| team | string | false | none | サーバー情報を表示できるロール/ユーザー名を定義します。 |
| is_remote_monitoring | boolean | false | none | このサーバーがリモートで監視されているかどうかを判断します。つまり、監視 pem エージェントが監視対象サーバーと同じホストにありません。 |
| tags | [object] | false | none | サーバーに関連付けられたタグのリスト。 |
| » text | string | false | none | タグの名前。 |
| » color | string | false | none | タグに関連付けられた色。 |
| replication_solution | string | false | none | サーバーが使用するレプリケーションソリューション |
| efm_cluster_name | string | false | none | EFM のクラスター名 |
| efm_service_name | string | false | none | EFM のサービス名 |
| efm_installation_path | string | false | none | EFM のインストールパス |
| patroni_cluster_name | string | false | none | Patroni のクラスター名 |
| patroni_installation_path | string | false | none | Patroni のインストールパス |
| patroni_config_path | string | false | none | Patroni の構成パス |
| comment | string | false | none | サーバーの説明 |
| username | string | false | none | データベースサーバーに接続するためのユーザー名 |
| gid | integer | false | none | サーバーグループの ID |
| hostaddr | string | false | none | サーバーホストアドレス |
| db_res | string | false | none | データベース制限 SQL 句 |
| role | string | false | none | 接続が使用するロール |
| agent_id | integer | false | none | サーバーにバインドされたエージェント ID |
| asb_host | string | false | none | PEM 接続の作成中にエージェントが使用する PEM データベースホスト |

次のページに続く

表 655 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| asb_port | integer | false | none | PEM 接続の作成中にエージェントが使用する PEM データベースポート |
| asb_username | string | false | none | PEM 接続の作成中にエージェントが使用する PEM データベースのユーザー名 |
| asb_database | string | false | none | エージェントが接続する PEM データベース |
| asb_sslmode | string | false | none | PEM データベース SSL モード |
| asb_password | string | false | none | 空の文字列 |
| agent_allowtakeover | boolean | false | none | サーバーを別のエージェントに引き継ぎできるかどうかを指定します。 |
| fgcolor | string | false | none | ブラウザツリー内のサーバーノードの前景色。デフォルトは黒#000000 |
| bgcolor | string | false | none | ブラウザツリーのサーバーノードの背景色。デフォルトは白#FFFFFF |
| sslcompression | boolean | false | none | SSL 経由で送信されるデータを圧縮するかどうかを決定します。 |
| use_ssh_tunnel | boolean | false | none | ssh トンネルを使用するかどうかを決定します。 |
| tunnel_host | string | false | none | トンネルホスト名 |
| tunnel_port | integer | false | none | トンネルポート番号 |
| tunnel_username | string | false | none | トンネルのユーザー名 |
| tunnel_password | string | false | none | トンネルパスワード |
| tunnel_authentication | boolean | false | none | トンネル認証を使用するかどうかを決定します。 |
| tunnel_identity_file | string | false | none | 認証に使用されるトンネル ID ファイル。 |
| tunnel_keep_alive | integer | false | none | ssh トンネルのキープアライブパケットを送信する秒単位の間隔。 |
| post_connection_sql | string | false | none | 接続の確立後に実行される SQL 。 |

### 列挙値

| Property | Value |
|---|---|
| replication_solution | none |
| replication_solution | efm |
| replication_solution | patroni |
| asb_sslmode | prefer |
| asb_sslmode | allow |
| asb_sslmode | disable |
| asb_sslmode | require |
| asb_sslmode | verify-ca |
| asb_sslmode | verify-full |

### ServerBodyUpdate


```
{
  "name": "string",
  "host": "string",
  "port": 0,
  "db": "string",
  "profile_id": 0,
  "serviceid": "string",
  "connection_params": {
    "added": [
      {
        "keyword": "hostaddr",
        "name": "hostaddr",
        "value": "qwde"
      }
    ],
    "changed": [
      {
        "keyword": "connect_timeout",
        "name": "connect_timeout",
        "value": "20"
      }
    ],
    "deleted": [
      {
        "keyword": "sslmode",
        "name": "sslmode",
        "value": "prefer"
      }
    ]
  },
```


(次のページに続く)

(前のページからの続き)


```
  "alert_blackout": true,
  "team": "string",
  "is_remote_monitoring": true,
  "tags": {
    "added": [
      {
        "color": "#0C767A",
        "text": "tag4"
      }
    ],
    "deleted": [
      {
        "text": "tag1",
        "color": "#8789AC",
        "old_text": "tag2"
      }
    ],
    "changed": [
      {
        "old_text": "tag",
        "text": "tag3"
      }
    ]
  },
  "replication_solution": "none",
  "efm_cluster_name": "string",
  "efm_service_name": "string",
  "efm_installation_path": "string",
  "patroni_cluster_name": "string",
  "patroni_installation_path": "string",
  "patroni_config_path": "string",
  "comment": "string",
  "username": "string",
  "gid": 0,
  "hostaddr": "string",
  "db_res": "string",
  "role": "string",
  "agent_id": 0,
  "asb_host": "string",
  "asb_port": 0,
  "asb_username": "string",
  "asb_database": "string",
  "asb_sslmode": "prefer",
  "asb_password": "string",
  "agent_allowtakeover": false,
```


(次のページに続く)

(前のページからの続き)


```
  "fgcolor": "#000000",
  "bgcolor": "#FFFFFF",
  "sslcompression": false,
  "use_ssh_tunnel": false,
  "tunnel_host": "string",
  "tunnel_port": 22,
  "tunnel_username": "string",
  "tunnel_password": "string",
  "tunnel_authentication": false,
  "tunnel_identity_file": "string",
  "tunnel_keep_alive": 0,
  "post_connection_sql": "string"
}
```


### プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| name | string | false | none | サーバー名 |
| host | string | false | none | サーバーホスト名またはアドレス |
| port | integer | false | none | サーバーポート |
| db | string | false | none | メンテナンスデータベース |
| profile_id | integer | false | none | プロファイル ID |
| serviceid | string | false | none | サーバーサービス ID |
| connection_params | [object] | false | none | サーバーの接続パラメーターのリスト。 |
| » name | string | false | none | パラメーター名 |
| » keyword | string | false | none | パラメーターキーワード |
| » value | any | false | none | パラメーター値 |
| alert_blackout | boolean | false | none | サーバー上のアラートがブラックアウトされているかどうかを判断します。 |
| team | string | false | none | サーバー情報を表示できるロール/ユーザー名を定義します。 |

次のページに続く

表 657 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| is_remote_monitoring | boolean | false | none | このサーバーがリモートで監視されているかどうかを判断します。つまり、監視 pem エージェントが監視対象サーバーと同じホストにありません。 |
| tags | [object] | false | none | サーバーに関連付けられたタグのリスト。 |
| » text | string | true | none | タグの名前。 |
| » color | string | true | none | タグに関連付けられた色。 |
| » old_text | string ¦ null | false | none | タグの以前の名前変更の追跡に使用されます。 |
| replication_solution | string | false | none | サーバーが使用するレプリケーションソリューション |
| efm_cluster_name | string | false | none | EFM のクラスター名 |
| efm_service_name | string | false | none | EFM のサービス名 |
| efm_installation_path | string | false | none | EFM のインストールパス |
| patroni_cluster_name | string | false | none | Patroni のクラスター名 |
| patroni_installation_path | string | false | none | Patroni のインストールパス |
| patroni_config_path | string | false | none | Patroni の構成パス |
| comment | string | false | none | サーバーの説明 |
| username | string | false | none | データベースサーバーに接続するためのユーザー名 |
| gid | integer | false | none | サーバーグループの ID |
| hostaddr | string | false | none | サーバーホストアドレス |
| db_res | string | false | none | データベース制限 SQL 句 |
| role | string | false | none | 接続が使用するロール |
| agent_id | integer | false | none | サーバーにバインドされたエージェント ID |
| asb_host | string | false | none | PEM 接続の作成中にエージェントが使用する PEM データベースホスト |

次のページに続く

表 657 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| asb_port | integer | false | none | PEM 接続の作成中にエージェントが使用する PEM データベースポート |
| asb_username | string | false | none | PEM 接続の作成中にエージェントが使用する PEM データベースのユーザー名 |
| asb_database | string | false | none | エージェントが接続する PEM データベース |
| asb_sslmode | string | false | none | PEM データベース SSL モード |
| asb_password | string | false | none | 空の文字列 |
| agent_allowtakeover | boolean | false | none | サーバーを別のエージェントに引き継ぎできるかどうかを指定します。 |
| fgcolor | string | false | none | ブラウザツリー内のサーバーノードの前景色。デフォルトは黒#000000 |
| bgcolor | string | false | none | ブラウザツリーのサーバーノードの背景色。デフォルトは白#FFFFFF |
| sslcompression | boolean | false | none | SSL 経由で送信されるデータを圧縮するかどうかを決定します。 |
| use_ssh_tunnel | boolean | false | none | ssh トンネルを使用するかどうかを決定します。 |
| tunnel_host | string | false | none | トンネルホスト名 |
| tunnel_port | integer | false | none | トンネルポート番号 |
| tunnel_username | string | false | none | トンネルのユーザー名 |
| tunnel_password | string | false | none | トンネルパスワード |
| tunnel_authentication | boolean | false | none | トンネル認証を使用するかどうかを決定します。 |
| tunnel_identity_file | string | false | none | 認証に使用されるトンネル ID ファイル。 |
| tunnel_keep_alive | integer | false | none | ssh トンネルのキープアライブパケットを送信する秒単位の間隔。 |
| post_connection_sql | string | false | none | 接続の確立後に実行される SQL 。 |

### 列挙値

| Property | Value |
|---|---|
| replication_solution | none |
| replication_solution | efm |
| replication_solution | patroni |
| asb_sslmode | prefer |
| asb_sslmode | allow |
| asb_sslmode | disable |
| asb_sslmode | require |
| asb_sslmode | verify-ca |
| asb_sslmode | verify-full |

### AgentBody

```
{
  "name": "string",
  "heartbeat_tolerance": 0,
  "alert_blackout": true,
  "profile_id": 0,
  "team": "string",
  "gid": 0,
  "tags": {
    "added": [
      {
        "color": "#0C767A",
        "text": "tag4"
      }
    ],
    "deleted": [
      {
        "text": "tag1",
        "color": "#8789AC",
        "old_text": "tag2"
      }
    ],
    "changed": [
      {
        "old_text": "tag",
        "text": "tag3"
      }
    ]
  },
  "ignore_mnt_points": ["string"]
}
```

エージェント情報、バージョン、機能、およびエージェントがインストールされているホストに関する詳細を含むオブジェクトのリストを返します。

例\

```
[{
  / *Agent information* /
  "id": 1,
  "gid": 0,
  "name": "Postgres Enterprise Manager Host"
  "version": "9.3.0",
  "active": True,
  "heartbeat_tolerance": 15,
  "alert_blackout": False,
  "owner": "postgres",
  "team": None,

  / *Deatils about the Host, on which it is installed* /
  "operating_sys": "Ubuntu 12.04.5 LTS",
  "host_name": "edb-ubuntu",
  "boot_time": "2016-09-19 12:28:27+05:30",
   "domain_name": "(none)",
   "window_domain": None,

  / *Agent capabilities* /
  "capability_list": "unix, getloadavg, disk_space, memory_usage, cpu_usage, network_
↪statistics, io_analysis, data_log_file_analysis, disk_busy_info,
os_statistics, os_info, slony_replication, auto_discover_servers, allow_server_restart
↪",

  / *Monitored servers* /
  "server_bind": "Postgres Enterprise Manager Server, PostgreSQL 11, PostgreSQL 12"
}]
```

ア

### プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| name | string | false | none | エージェントの説明名前。 |

次のページに続く

表 659 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| heartbeat_tolerance | integer | false | none | エージェントのハートビート許容秒単位。ハートビートがどれだけ遅れた場合、逃したと見なされるかを指定します。 |
| alert_blackout | boolean | false | none | エージェントのアラートをブラックアウトするかどうかを判断します。 |
| profile_id | integer | false | none | このエージェントを指定されたプロファイル ID で管理する必要があるかどうかを判断します。 |
| team | string | false | none | このエージェントを表示できるロール/ユーザーの名前。 null/空の値は、すべてのユーザーがこのエージェントを表示できることを示します。 |
| gid | integer | false | none | グループの ID |
| tags | [object] | false | none | エージェントに関連付けられたタグのリスト。 |
| » text | string | true | none | タグの名前。 |
| » color | string | true | none | タグに関連付けられた色。 |
| » old_text | string ¦ null | false | none | タグの以前の名前変更の追跡に使用されます。 |
| ignore_mnt_points | [string] | false | none | ディスク使用量の計算でマウントポイントを無視するために使用されます |

### AgentJobBodyGet

```
{
  "jobname": "string",
  "jobenabled": true,
  "jobdesc": "string"
}
```

プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| jobname | string | false | none | ジョブの名前。 |
| jobenabled | boolean | false | none | ジョブが有効か無効かを判断します。 |
| jobdesc | string | false | none | ジョブの説明。 |

### AgentJobBody


```
{
  "jobid": 0,
  "jobname": "string",
  "jobenabled": true,
  "jobdesc": "string",
  "jobcreated": "2019-08-24T14:15:22Z",
  "jobchanged": "2019-08-24T14:15:22Z",
  "jlgstatus": "string",
  "jobnextrun": "2019-08-24T14:15:22Z",
  "joblastrun": "2019-08-24T14:15:22Z",
  "is_alert_job": true,
  "notify": "string",
  "email_group_id": "string",
  "jsteps": [],
  "jschedules": [
    {
      "jscid": 0,
      "jscjobid": 0,
      "jscname": "string",
      "jscdesc": "string",
      "jscenabled": true,
      "jscstart": "2019-08-24T14:15:22Z",
      "jscend": "2019-08-24T14:15:22Z",
      "jscweekdays": [false, false, false, false, false, true, false],
      "jscmonthdays": [
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        true,
        false,
```


(次のページに続く)

(前のページからの続き)

```
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false
      ],
      "jscmonths": [
        true,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false
      ],
      "jschours": [
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
```

(次のページに続く)

(前のページからの続き)


```
      false,
      false,
      false,
      false,
      false,
      false,
      false,
      false,
      false,
      true,
      false,
      false,
      false,
      false,
      false,
      false,
      false
    ],
    "jscminutes": [
      false,
      false,
      false,
      false,
      true,
      false,
      false,
      false,
      false,
      false,
      false,
      false,
      false,
      false,
      false,
      false,
      false,
      false,
      false,
      false,
      false,
      false,
      false,
      false,
      false,
      false,
```


(次のページに続く)

(前のページからの続き)


```
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false,
        false
      ],
      "jscexceptions": [null],
      "jsctimezone": "string"
    }
  ]
}
```

プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| jobid | integer | false | none | ジョブの ID。 |
| jobname | string | false | none | ジョブの名前。 |
| jobenabled | boolean | false | none | ジョブが有効か無効かを判断します。 |
| jobdesc | string | false | none | ジョブの説明。 |
| jobcreated | string(date-time) | false | none | ジョブが作成されたときのタイムスタンプ。 |
| jobchanged | string(date-time) | false | none | ジョブが最後に変更されたときのタイムスタンプ。 |
| jlgstatus | string | false | none | 最後にジョブ実行のステータス。 |
| jobnextrun | string(date-time) ¦ null | false | none | 次にスケジュールされた実行のタイムスタンプ。 |
| joblastrun | string(date-time) ¦ null | false | none | 最終実行のタイムスタンプ。 |
| is_alert_job | boolean | false | none | ジョブがアラートジョブであるかどうかを示します。 |
| notify | string | false | none | ジョブの通知設定。 |
| email_group_id | string | false | none | 通知用のメールグループの ID。 |
| jsteps | [object] | false | none | ジョブのステップのリスト。 |
| » jstid | integer | false | none | ステップの ID。 |
| » jstjobid | integer | false | none | ステップに関連付けられたジョブの ID。 |
| » jstname | string | false | none | ステップの名前。 |
| » jstdesc | string | false | none | ステップの説明。 |
| » jstenabled | boolean | false | none | ステップが有効か無効かを決定します。 |
| » jstkind | any | false | none | ステップの種類を決定します。ブール値 true=SQL、false=Batch または文字列's'=SQL、'b'=Batch、'r'=Report を受け入れます。 |

oneOf

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » _anonymous_ | boolean | false | none | なし |

xor

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » _anonymous_ | string | false | none | なし |

続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » jstcode | string | false | none | ステップのコードまたはスクリプト。 |
| » server_id | integer | false | none | ステップに関連付けられたサーバーの ID。 |
| » database_name | string | false | none | ステップに関連付けられたデータベースの名前。 |
| » jstonerror | string | false | none | ステップ'Fail'、'Success'、'Ignore' の実行中にエラーが発生した場合のエージェントの動作を指定します。 |
| » report_template_id | integer ¦ null | false | none | レポートテンプレートの ID。 jstkind が'r' レポートステップの場合に必要です。 |
| jschedules | [object] | false | none | ジョブのスケジュールのリスト。 |
| » jscid | integer ¦ null | false | none | スケジュールの ID。 |
| » jscjobid | integer ¦ null | false | none | スケジュールに関連付けられたジョブの ID。 |
| » jscname | string | false | none | スケジュールの名前。 |
| » jscdesc | string | false | none | スケジュールの説明。 |
| » jscenabled | boolean | false | none | スケジュールが有効か無効かを決定します。 |
| » jscstart | string(date-time) | false | none | スケジュールの開始日時。 |
| » jscend | string(date-time) | false | none | スケジュールの終了日時。 |

次のページに続く

表 664 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » jscweekdays | [boolean] | false | none | 曜日の 7 要素のブール配列。インデックス 0=日曜日、1=月曜日、2=火曜日、3=水曜日、4=木曜日、5=金曜日、6=土曜日。インデックスで true は、スケジュールがその日に実行されることを意味します。 |
| » jscmonthdays | [boolean] | false | none | 日付の 32 要素のブール配列。インデックス 0=1 日、インデックス 30=31 日、インデックス 31=月末日。インデックスで true は、スケジュールがその日に実行されることを意味します。 |
| » jscmonths | [boolean] | false | none | 月の 12 要素のブール配列。インデックス 0=1 月、1=2 月、...、11=12 月。インデックスで true は、スケジュールがその月に実行されることを意味します。 |
| » jschours | [boolean] | false | none | その日の時間 0〜23 を表す 24 要素のブール配列。インデックス 0=真夜中、インデックス 12=正午、インデックス 23=午後 11 時。インデックスで true は、スケジュールがその時間に実行されることを意味します。 |
| » jscminutes | [boolean] | false | none | 時間の分 0〜59 の 60 要素のブール配列。インデックスで true は、スケジュールがその分に実行されることを意味します。 |
| » jscexceptions | [any] | false | none | スケジュールの例外のリスト。 |
| » jsctimezone | string | false | none | スケジュールのタイムゾーン。 |

### 列挙値

| Property | Value |
|---|---|
| _anonymous_ | s |
| _anonymous_ | b |
| _anonymous_ | r |

### AgentJobBodyPut

```
{
  "jobname": "string",
  "jobenabled": true,
  "jobdesc": "string",
  "jsteps": {
    "added": [
      {
        "jstname": "string",
        "jstdesc": "string",
        "jstenabled": true,
        "jstkind": true,
        "jstcode": "string",
        "server_id": 0,
        "database_name": "string",
        "jstonerror": "f",
        "report_template_id": 0
      }
    ],
    "changed": [
      {
        "jstid": 0,
        "jstname": "string",
        "jstdesc": "string",
        "jstenabled": true,
        "jstkind": true,
        "jstcode": "string",
        "server_id": 0,
        "database_name": "string",
        "jstonerror": "f",
        "report_template_id": 0,
        "jstnextrun": "string"
      }
    ]
  },
  "jschedules": {
    "added": [
```

(次のページに続く)

(前のページからの続き)


```
    {
      "jscname": "string",
      "jscdesc": "string",
      "jscenabled": true,
      "jscstart": "2023-10-09 22:43 +05:30",
      "jscend": "2023-10-11 22:43 +05:30",
      "jscweekdays": ["Friday"],
      "jscmonthdays": [1, 15],
      "jscmonths": [
        "January",
        "February",
        "March",
        "April",
        "May",
        "June",
        "July",
        "August",
        "September",
        "October",
        "November",
        "December"
      ],
      "jschours": [16],
      "jscminutes": [0, 30],
      "jscexceptions": [
        {
          "jexdate": "2023-10-19",
          "jextime": "09:39:00"
        }
      ],
      "jsctimezone": "America/New_York"
    }
  ],
  "changed": [
    {
      "jscid": 0,
      "jscname": "string",
      "jscdesc": "string",
      "jscenabled": true,
      "jscstart": "2023-10-09 22:43 +05:30",
      "jscend": "2023-10-11 22:43 +05:30",
      "jscweekdays": ["Friday"],
      "jscmonthdays": [1, 15],
      "jscmonths": [
        "January",
```


(次のページに続く)

(前のページからの続き)


```
          "February",
          "March",
          "April",
          "May",
          "June",
          "July",
          "August",
          "September",
          "October",
          "November",
          "December"
        ],
        "jschours": [16],
        "jscminutes": [0, 30],
        "jscexceptions": [
          {
            "jexdate": "2023-10-19",
            "jextime": "09:39:00"
          }
        ],
        "jsctimezone": "America/New_York"
      }
    ]
  },
  "notify": "DEFAULT",
  "email_group_id": "string"
}
```


### プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| jobname | string | false | none | ジョブの名前。 |
| jobenabled | boolean | false | none | ジョブが有効か無効かを判断します。 |
| jobdesc | string | false | none | ジョブの説明。 |
| jsteps | object | false | none | なし |
| » added | [object] | false | none | 追加されたステップのリスト。 |
| » jstname | string | false | none | ステップの名前。 |
| » jstdesc | string | false | none | ステップの説明。 |
| » jstenabled | boolean | false | none | ステップが有効か無効かを決定します。 |

次のページに続く

表 666 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » jstkind | any | false | none | ステップの種類を決定します。ブール値 true=SQL、false=Batch または文字列's'= SQL、'b'=Batch、'r'=Report を受け入れます。 |

oneOf

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » _anonymous_ | boolean | false | none | なし |

xor

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » _anonymous_ | string | false | none | なし |

続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » jstcode | string | false | none | ステップのコードまたはスクリプト。 |
| » server_id | integer | false | none | ステップに関連付けられたサーバーの ID。 |
| » database_name | string | false | none | ステップに関連付けられたデータベースの名前。 |
| » jstonerror | string | false | none | ステップ'Fail'、'Success'、'Ignore' の実行中にエラーが発生した場合のエージェントの動作を指定します。 |
| » report_template_id | integer | false | none | レポートテンプレートの ID。 jstkind が'r' レポートステップの場合に必要です。 |
| » changed | [object] | false | none | 変更されたステップのリスト。 |
| » jstid | integer | false | none | ステップの ID。 |
| » jstname | string | false | none | ステップの名前。 |

次のページに続く

表 669 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » jstdesc | string | false | none | ステップの説明。 |
| » jstenabled | boolean | false | none | ステップが有効か無効かを決定します。 |
| » jstkind | any | false | none | ステップの種類を決定します。ブール値 true=SQL、false=Batch または文字列's'= SQL、'b'=Batch、 'r'=Report を受け入れます。 |

oneOf

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » _anonymous_ | boolean | false | none | なし |

xor

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » _anonymous_ | string | false | none | なし |

続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » jstcode | string | false | none | ステップのコードまたはスクリプト。 |
| » server_id | integer | false | none | ステップに関連付けられたサーバーの ID。 |
| » database_name | string | false | none | ステップに関連付けられたデータベースの名前。 |
| » jstonerror | string | false | none | ステップ'Fail'、'Success'、'Ignore' の実行中にエラーが発生した場合のエージェントの動作を指定します。 |
| » report_template_id | integer | false | none | レポートテンプレートの ID。 jstkind が'r' レポートステップの場合に必要です。 |
| » jstnextrun | string | false | none | なし |

次のページに続く

表 672 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| jschedules | object | false | none | なし |
| » added | [object] | false | none | 追加されたスケジュールのリスト。 |
| » jscname | string | false | none | スケジュールの名前。 |
| » jscdesc | string | false | none | スケジュールの説明。 |
| » jscenabled | boolean | false | none | スケジュールが有効か無効かを決定します。 |
| » jscstart | string | false | none | スケジュールの開始日時 YYYY-MM-DD HH:mm ±HH:mm 。 |
| » jscend | string | false | none | スケジュールの終了日時 YYYY-MM-DD HH:mm ±HH:mm 。 |
| » jscweekdays | [string] | false | none | スケジュールを実行する曜日の配列。必要な日のみを渡します。 |
| » jscmonthdays | [integer] | false | none | スケジュールを実行する日の番号 1〜31 の配列。 ' 月の最終日' には 32 を使用します。必要な日のみを渡します。 |
| » jscmonths | [string] | false | none | スケジュールを実行する月の名前の配列。必要な月のみを渡します。 |
| » jschours | [integer] | false | none | スケジュールを実行する時間 0〜23 の配列。 0=真夜中、12=正午、23=午後 11 時。必要な時間だけを渡します。 |
| » jscminutes | [integer] | false | none | スケジュールを実行する分の分配列 0〜59。必要な分だけを渡します。 |
| » jscexceptions | [object] | false | none | スケジュールの例外のリスト。 |
| » jexdate | string(date) | false | none | スケジュール例外の日付。 |
| » jextime | string(time) | false | none | スケジュール例外の時間。 |
| » jsctimezone | string | false | none | スケジュールのタイムゾーン。標準の IANA タイムゾーン名を使用します。 |

次のページに続く

表 672 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » changed | [object] | false | none | 変更されたスケジュールのリスト。 |
| » jscid | integer ¦ null | false | none | スケジュールの ID。 |
| » jscname | string | false | none | スケジュールの名前。 |
| » jscdesc | string | false | none | スケジュールの説明。 |
| » jscenabled | boolean | false | none | スケジュールが有効か無効かを決定します。 |
| » jscstart | string | false | none | スケジュールの開始日時 YYYY-MM-DD HH:mm ±HH:mm 。 |
| » jscend | string | false | none | スケジュールの終了日時 YYYY-MM-DD HH:mm ±HH:mm 。 |
| » jscweekdays | [string] | false | none | スケジュールを実行する曜日の配列。必要な日のみを渡します。 |
| » jscmonthdays | [integer] | false | none | スケジュールを実行する日の番号 1〜31 の配列。 ' 月の最終日' には 32 を使用します。必要な日のみを渡します。 |
| » jscmonths | [string] | false | none | スケジュールを実行する月の名前の配列。必要な月のみを渡します。 |
| » jschours | [integer] | false | none | スケジュールを実行する時間 0〜23 の配列。 0=真夜中、12=正午、23=午後 11 時。必要な時間だけを渡します。 |
| » jscminutes | [integer] | false | none | スケジュールを実行する分の分配列 0〜59。必要な分だけを渡します。 |
| » jscexceptions | [object] | false | none | スケジュールの例外のリスト。 |
| » jexdate | string(date) | false | none | スケジュール例外の日付。 |
| » jextime | string(time) | false | none | スケジュール例外の時間。 |
| » jsctimezone | string | false | none | スケジュールのタイムゾーン。標準の IANA タイムゾーン名を使用します。 |
| notify | string | false | none | ジョブの通知設定。 |

次のページに続く

表 672 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| email_group_id | string | false | none | 通知用のメールグループの ID。 |

### 列挙値

| Property | Value |
|---|---|
| _anonymous_ | s |
| _anonymous_ | b |
| _anonymous_ | r |
| jstonerror | f |
| jstonerror | s |
| jstonerror | i |
| _anonymous_ | s |
| _anonymous_ | b |
| _anonymous_ | r |
| jstonerror | f |
| jstonerror | s |
| jstonerror | i |
| notify | DEFAULT |
| notify | ON_FAILURE |
| notify | ALWAYS |
| notify | NEVER |

## AgentJobBodyPost

```
{
  "jobname": "string",
  "jobenabled": true,
  "jobdesc": "string",
  "jsteps": [],
  "jschedules": [
    {
      "jscname": "string",
      "jscdesc": "string",
      "jscenabled": true,
      "jscstart": "2023-10-09 22:43 +05:30",
      "jscend": "2023-10-11 22:43 +05:30",
      "jscweekdays": ["Friday"],
      "jscmonthdays": [1, 15],
      "jscmonths": [
        "January",
        "February",
```

(次のページに続く)

(前のページからの続き)


```
        "March",
        "April",
        "May",
        "June",
        "July",
        "August",
        "September",
        "October",
        "November",
        "December"
      ],
      "jschours": [16],
      "jscminutes": [0, 30],
      "jscexceptions": [
        {
          "jexdate": "2023-10-19",
          "jextime": "09:39:00"
        }
      ],
      "jsctimezone": "America/New_York"
    }
  ],
  "notify": "DEFAULT",
  "email_group_id": "string"
}
```


### プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| jobname | string | false | none | ジョブの名前。 |
| jobenabled | boolean | false | none | ジョブが有効か無効かを判断します。 |
| jobdesc | string | false | none | ジョブの説明。 |
| jsteps | [object] | false | none | ジョブのステップのリスト。 |
| » jstname | string | false | none | ステップの名前。 |
| » jstdesc | string | false | none | ステップの説明。 |
| » jstenabled | boolean | false | none | ステップが有効か無効かを決定します。 |

次のページに続く

表 674 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » jstkind | any | false | none | ステップの種類を決定します。ブール値 true=SQL、false=Batch または文字列's'= SQL、'b'=Batch、'r'=Report を受け入れます。 |

oneOf

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » _anonymous_ | boolean | false | none | なし |

xor

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » _anonymous_ | string | false | none | なし |

続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » jstcode | string | false | none | ステップのコードまたはスクリプト。 |
| » server_id | integer | false | none | ステップに関連付けられたサーバーの ID。 |
| » database_name | string | false | none | ステップに関連付けられたデータベースの名前。 |
| » jstonerror | string | false | none | ステップ'Fail'、'Success'、'Ignore' の実行中にエラーが発生した場合のエージェントの動作を指定します。 |
| » report_template_id | integer | false | none | レポートテンプレートの ID。 jstkind が'r' レポートステップの場合に必要です。 |
| jschedules | [object] | false | none | ジョブのスケジュールのリスト。 |
| » jscname | string | false | none | スケジュールの名前。 |
| » jscdesc | string | false | none | スケジュールの説明。 |

次のページに続く

表 677 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| » jscenabled | boolean | false | none | スケジュールが有効か無効かを決定します。 |
| » jscstart | string | false | none | スケジュールの開始日時　YYYY-MM-DD HH:mm ±HH:mm 。 |
| » jscend | string | false | none | スケジュールの終了日時　YYYY-MM-DD HH:mm ±HH:mm 。 |
| » jscweekdays | [string] | false | none | スケジュールを実行する曜日の配列。必要な日のみを渡します。 |
| » jscmonthdays | [integer] | false | none | スケジュールを実行する日の番号 1〜31 の配列。 ' 月の最終日' には 32 を使用します。必要な日のみを渡します。 |
| » jscmonths | [string] | false | none | スケジュールを実行する月の名前の配列。必要な月のみを渡します。 |
| » jschours | [integer] | false | none | スケジュールを実行する時間 0〜23 の配列。 0=真夜中、12=正午、23=午後 11 時。必要な時間だけを渡します。 |
| » jscminutes | [integer] | false | none | スケジュールを実行する分の分配列 0〜59。必要な分だけを渡します。 |
| » jscexceptions | [object] | false | none | スケジュールの例外のリスト。 |
| » jexdate | string(date) | false | none | スケジュール例外の日付。 |
| » jextime | string(time) | false | none | スケジュール例外の時間。 |
| » jsctimezone | string | false | none | スケジュールのタイムゾーン。標準の IANA タイムゾーン名を使用します。 |
| notify | string | false | none | ジョブの通知設定。 |
| email_group_id | string | false | none | 通知用のメールグループの ID。 |

### 列挙値

| Property | Value |
|---|---|
| _anonymous_ | s |
| _anonymous_ | b |
| _anonymous_ | r |
| jstonerror | f |
| jstonerror | s |
| jstonerror | i |
| notify | DEFAULT |
| notify | ON_FAILURE |
| notify | ALWAYS |
| notify | NEVER |

### データベース

```
{
  "database_name": "string",
  "connections_allowed": true,
  "system_database": true,
  "recorded_time": "2019-08-24T14:15:22Z",
  "encoding": "string",
  "server_id": 0
}
```

### プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| database_name | string | false | none | データベース名 |
| connections_allowed | boolean | false | none | データベースへの接続を許可するかどうかを決定します。 |
| system_database | boolean | false | none | データベースがシステムデータベースであるかどうかを判断します。 |
| recorded_time | string(date-time) | false | none | データベースが PEM エージェントによって最初に検出されたときを指定します。 |
| encoding | string | false | none | データベースエンコーディング |

次のページに続く

表 679 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| server_id | integer | false | none | データベースのサーバー ID |

### スキーマ

```
{
  "schema_name": "string",
  "database_name": "string",
  "server_id": 0,
  "recorded_time": "2019-08-24T14:15:22Z"
}
```

### プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| schema_name | string | false | none | スキーマ名 |
| database_name | string | false | none | スキーマのデータベース名 |
| server_id | integer | false | none | サーバー ID スキーマ |
| recorded_time | string(date-time) | false | none | スキーマが PEM エージェントによって最初に検出されたときを指定します。 |

### テーブル

```
{
  "table_name": "string",
  "schema_name": "string",
  "database_name": "string",
  "server_id": 0,
  "has_primary_key": true,
  "recorded_time": "string"
}
```

### プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| table_name | string | false | none | テーブル名 |

次のページに続く

表 681 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| schema_name | string | false | none | スキーマ名 |
| database_name | string | false | none | データベース名 |
| server_id | integer | false | none | サーバー ID |
| has_primary_key | boolean | false | none | テーブルに主キーがあるかどうかを判断します。 |
| recorded_time | string | false | none | テーブルが PEM エージェントによって最初に検出されたときを指定します。 |

### インデックス

```
{
  "index_name": "string",
  "ind_keys": [0],
  "table_name": "string",
  "schema_name": "string",
  "database_name": "string",
  "server_id": 0,
  "recorded_time": "string"
}
```

### プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| index_name | string | false | none | インデックス名 |
| ind_keys | [integer] | false | none | テーブル内のインデックス付き列の位置 |
| table_name | string | false | none | テーブル名 |
| schema_name | string | false | none | スキーマ |
| database_name | string | false | none | データベース名 |
| server_id | integer | false | none | サーバー ID |
| recorded_time | string | false | none | インデックスが PEM エージェントによって最初に検出されたときを指定します。 |

### シークエンス

```
{
  "sequence_name": "string",
  "schema_name": "string",
  "database_name": "string",
  "server_id": 0,
  "recorded_time": "2019-08-24T14:15:22Z"
}
```

### プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| sequence_name | string | false | none | シークエンス名 |
| schema_name | string | false | none | スキーマ名 |
| database_name | string | false | none | データベース名 |
| server_id | integer | false | none | サーバー ID |
| recorded_time | string(date-time) | false | none | シーケンスが PEM エージェントによって最初に検出されたときを指定します。 |

### ファンクション

```
{
  "function_name": "string",
  "function_type": "string",
  "return_type": "string",
  "arg_types": ["string"],
  "function_binary": "string",
  "extension_name": "string",
  "package_name": "string",
  "schema_name": "string",
  "database_name": "string",
  "server_id": 0,
  "recorded_time": "2019-08-24T14:15:22Z"
}
```

### プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| function_name | string | false | none | ファンクション名 |

次のページに続く

表 684 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| function_type | string | false | none | ファンクションタイプ |
| return_type | string | false | none | ファンクションの戻り値の型 |
| arg_types | [string] | false | none | ファンクションの引数の型 |
| function_binary | string | false | none | なし |
| extension_name | string | false | none | なし |
| package_name | string | false | none | ファンクションパッケージ名 |
| schema_name | string | false | none | ファンクションスキーマ |
| database_name | string | false | none | ファンクションデータベース名 |
| server_id | integer | false | none | ファンクションサーバー ID |
| recorded_time | string(date-time) | false | none | ファンクションが PEM エージェントによって最初に検出されたときを指定します。 |

### ビュー


```
{
  "view_name": "string",
  "view_type": "string",
  "ispopulated": true,
  "view_owner": "string",
  "definition": "string",
  "tablespace_name": "string",
  "schema_name": "string",
  "database_name": "string",
  "server_id": 0,
  "recorded_time": "string"
}
```


### プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| view_name | string | false | none | ビュー名 |
| view_type | string | false | none | ビュータイプ |
| ispopulated | boolean | false | none | ビューが設定されているかどうかを判断します。 |

次のページに続く

表 685 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| view_owner | string | false | none | ビューの所有者 |
| definition | string | false | none | ビュー定義 |
| tablespace_name | string | false | none | ビューテーブルスペース |
| schema_name | string | false | none | ビュースキーマ |
| database_name | string | false | none | データベースを見る |
| server_id | integer | false | none | ビューサーバー ID |
| recorded_time | string | false | none | ビューが PEM エージェントによって最初に検出されたときを指定します。 |

### Webhook

```
{
  "id": 0,
  "name": "string",
  "url": "string",
  "enabled": true,
  "method": "POST",
  "http_headers": [
    {
      "http_header_id": 0,
      "http_header_key": "string",
      "http_header_value": "string"
    }
  ],
  "payload_template": "{text: You have new alert from PEM, attachments: [{text:␣
↪AlertID \n%AlertID% \nAlertName \n%AlertName%}]}",
  "low_alert": true,
  "med_alert": true,
  "high_alert": true,
  "cleared_alert": true,
  "payload_type": "ALERT"
}
```

#### プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| id | integer | false | none | Webhook ID |
| name | string | false | none | Webhook 名 |
| url | string | false | none | Webhook URL |

次のページに続く

表 686 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| enabled | boolean | false | none | Webhook が有効になっているかどうかを判断します。 |
| method | string | false | none | リクエストメソッド |
| http_headers | *Platform Compatibility* ] | false | none | Webhook で使用される HTTP ヘッダーのキー/値ペアのリスト |
| payload_template | string | false | none | ペイロードテンプレート。以下は、Slack のペイロードの例です。 |
| low_alert | boolean | false | none | 低アラートを有効にするかどうかを決定します。 |
| med_alert | boolean | false | none | 中程度のアラートを有効にするかどうかを決定します。 |
| high_alert | boolean | false | none | 厳重なアラートを有効にするかどうかを決定します。 |
| cleared_alert | boolean | false | none | クリアされたアラートが有効かどうかを判断します。 |
| payload_type | string | false | none | ペイロードタイプ |

### 列挙値

| Property | Value |
|---|---|
| method | POST |
| method | PUT |
| payload_type | ALERT |
| payload_type | JOB |

### WebhookPostData

```
{
  "name": "string",
  "url": "string",
  "enabled": true,
  "method": "POST",
  "http_headers": [
    {
      "http_header_key": "string",
      "http_header_value": "string"
```

(次のページに続く)

(前のページからの続き)

```
    }
  ],
  "payload_template": "{ Alert Name:%AlertName%,type:Webhook }",
  "low_alert": true,
  "med_alert": true,
  "high_alert": true,
  "cleared_alert": true,
  "payload_type": "ALERT"
}
```

### プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| name | string | false | none | Webhook 名 |
| url | string | false | none | Webhook URL |
| enabled | boolean | false | none | Webhook が有効になっているかどうかを判断します。 |
| method | string | false | none | リクエストメソッド |
| http_headers | *Platform Compatibility* ] | false | none | Webhook で使用される HTTP ヘッダーのキー/値ペアのリスト |
| payload_template | string | false | none | ペイロードテンプレート |
| low_alert | boolean | false | none | 低アラートを有効にするかどうかを決定します。 |
| med_alert | boolean | false | none | 中程度のアラートを有効にするかどうかを決定します。 |
| high_alert | boolean | false | none | 厳重なアラートを有効にするかどうかを決定します。 |
| cleared_alert | boolean | false | none | クリアされたアラートが有効かどうかを判断します。 |
| payload_type | string | false | none | ペイロードタイプ |

### 列挙値

| Property | Value |
|---|---|
| method | POST |
| method | PUT |

次のページに続く

表 689 – 前のページからの続き

| Property | Value |
|---|---|
| payload_type | ALERT |
| payload_type | JOB |

### WebhookPutData

```
{
  "name": "string",
  "url": "string",
  "enabled": true,
  "method": "POST",
  "http_headers": {
    "added": [
      {
        "http_header_key": "string",
        "http_header_value": "string"
      }
    ],
    "changed": [
      {
        "http_header_id": 0,
        "http_header_key": "string",
        "http_header_value": "string"
      }
    ],
    "deleted": [
      {
        "http_header_id": 0
      }
    ],
    "payload_type": "ALERT"
  },
  "payload_template": "{text: You have new alert from PEM, attachments: [{text:␣
↪AlertID \n%AlertID% \nAlertName \n%AlertName%}]}",
  "low_alert": true,
  "med_alert": true,
  "high_alert": true,
  "cleared_alert": true
}
```

### プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| name | string | false | none | Webhook 名 |
| url | string | false | none | Webhook URL |
| enabled | boolean | false | none | Webhook が有効になっているかどうかを判断します。 |
| method | string | false | none | リクエストメソッド |
| http_headers | object | false | none | なし |
| » added | *Platform Compatibility* ] | false | none | 追加する HTTP ヘッダーキー/値ペアのリスト |
| » changed | *Platform Compatibility* ] | false | none | 更新する HTTP ヘッダーのキー/値ペアのリスト |
| » deleted | [object] | false | none | 削除する HTTP ヘッダー ID のリスト |
| » http_header_id | integer | false | none | HTTP ヘッダーの ID |
| » payload_type | string | false | none | ペイロードタイプ |
| payload_template | string | false | none | ペイロードテンプレート。以下は、Slack のペイロードの例です。 |
| low_alert | boolean | false | none | 低アラートを有効にするかどうかを決定します。 |
| med_alert | boolean | false | none | 中程度のアラートを有効にするかどうかを決定します。 |
| high_alert | boolean | false | none | 厳重なアラートを有効にするかどうかを決定します。 |
| cleared_alert | boolean | false | none | クリアされたアラートが有効かどうかを判断します。 |

### 列挙値

| Property | Value |
|---|---|
| method | POST |
| method | PUT |
| payload_type | ALERT |
| payload_type | JOB |

### WebhookGetHTTPHeaders

```
{
  "http_header_id": 0,
  "http_header_key": "string",
  "http_header_value": "string"
}
```

#### プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| http_header_id | integer | false | none | HTTP ヘッダーの ID |
| http_header_key | string | false | none | HTTP ヘッダーキーの名前 |
| http_header_value | string | false | none | HTTP ヘッダーキーの名前 |

### WebhookHTTP ヘッダー

```
{
  "http_header_key": "string",
  "http_header_value": "string"
}
```

#### プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| http_header_key | string | false | none | HTTP ヘッダーキーの名前 |
| http_header_value | string | false | none | HTTP ヘッダーキーの名前 |

### メールグループ

```
{
  "id": 0,
  "oid": 0,
  "gid": 0,
  "name": "string",
  "to_addr": "string",
  "cc_addr": "string",
```

(次のページに続く)

(前のページからの続き)

```
  "bcc_addr": "string",
  "from_addr": "string",
  "reply_to_addr": "string",
  "subject_prefix": "string",
  "from_time": "00:00:00",
  "to_time": "23:59:59"
}
```

### プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| id | integer | false | none | メールグループ ID |
| oid | integer | false | none | メールグループオプション ID |
| gid | integer | false | none | メールグループ ID |
| name | string | false | none | メールグループ名 |
| to_addr | string | false | none | メールグループオプションの受信者アドレス |
| cc_addr | string | false | none | メールグループオプション CC アドレス |
| bcc_addr | string | false | none | メールグループオプション BCC アドレス |
| from_addr | string | false | none | メールグループオプションの送信者アドレス |
| reply_to_addr | string | false | none | メールグループオプションの返信先アドレス |
| subject_prefix | string | false | none | メールグループオプションの件名プレフィックス |
| from_time | string | false | none | メンバーの時間から選択した時間範囲までのメールグループオプション |
| to_time | string | false | none | メンバーの時間範囲を選択するメールグループオプション |

## EmailGroupsPostData

```
{
  "name": "string",
  "options": [
    {
      "to_addr": "string",
      "cc_addr": "string",
```

(次のページに続く)

(前のページからの続き)


```
      "bcc_addr": "string",
      "from_addr": "string",
      "reply_to_addr": "string",
      "subject_prefix": "string",
      "from_time": "00:00:00",
      "to_time": "23:59:59"
    }
  ]
}
```


プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| name | string | false | none | メールグループ名 |
| options | [Email-GroupOptions] | false | none | メールグループで使用されるメールグループオプションのリスト |

### EmailGroupsPutData


```
{
  "name": "string",
  "options": {
    "added": [
      {
        "to_addr": "string",
        "cc_addr": "string",
        "bcc_addr": "string",
        "from_addr": "string",
        "reply_to_addr": "string",
        "subject_prefix": "string",
        "from_time": "00:00:00",
        "to_time": "23:59:59"
      }
    ],
    "changed": [
      {
        "oid": 0,
        "to_addr": "string",
        "cc_addr": "string",
        "bcc_addr": "string",
        "from_addr": "string",
        "reply_to_addr": "string",
```


(次のページに続く)

(前のページからの続き)


```
        "subject_prefix": "string",
        "from_time": "00:00:00",
        "to_time": "23:59:59"
      }
    ],
    "deleted": [
      {
        "oid": 0
      }
    ]
  }
}
```


### プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| name | string | false | none | メールグループ名 |
| options | object | false | none | なし |
| » added | [EmailGroupOptions] | false | none | 追加するメールグループオプション |
| » changed | [EmailGroupOptionsChanged] | false | none | 更新するメールグループオプション。 |
| » deleted | [object] | false | none | 削除する HTTP ヘッダー ID のリスト |
| » oid | integer | false | none | メールグループオプション ID |

### EmailGroupOptions


```
{
  "to_addr": "string",
  "cc_addr": "string",
  "bcc_addr": "string",
  "from_addr": "string",
  "reply_to_addr": "string",
  "subject_prefix": "string",
  "from_time": "00:00:00",
  "to_time": "23:59:59"
}
```

プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| to_addr | string | false | none | メールグループオプションの受信者アドレス |
| cc_addr | string | false | none | メールグループオプション CC アドレス |
| bcc_addr | string | false | none | メールグループオプション BCC アドレス |
| from_addr | string | false | none | メールグループオプションの送信者アドレス |
| reply_to_addr | string | false | none | メールグループオプションの返信先アドレス |
| subject_prefix | string | false | none | メールグループオプションの件名プレフィックス |
| from_time | string | false | none | メンバーの時間から選択した時間範囲までのメールグループオプション |
| to_time | string | false | none | メンバーの時間範囲を選択するメールグループオプション |

### EmailGroupOptionsChanged

```
{
  "oid": 0,
  "to_addr": "string",
  "cc_addr": "string",
  "bcc_addr": "string",
  "from_addr": "string",
  "reply_to_addr": "string",
  "subject_prefix": "string",
  "from_time": "00:00:00",
  "to_time": "23:59:59"
}
```

プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| oid | integer | false | none | メールグループオプション ID |
| to_addr | string | false | none | メールグループオプションの受信者アドレス |

次のページに続く

表 698 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| cc_addr | string | false | none | メールグループオプション CC アドレス |
| bcc_addr | string | false | none | メールグループオプション BCC アドレス |
| from_addr | string | false | none | メールグループオプションの送信者アドレス |
| reply_to_addr | string | false | none | メールグループオプションの返信先アドレス |
| subject_prefix | string | false | none | メールグループオプションの件名プレフィックス |
| from_time | string | false | none | メンバーの時間から選択した時間範囲までのメールグループオプション |
| to_time | string | false | none | メンバーの時間範囲を選択するメールグループオプション |

### ExcludeDatabases

```
{
  "server_id": 0,
  "name": "string",
  "exclude_databases": ["string"]
}
```

#### プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| server_id | integer | false | none | サーバー ID |
| name | string | false | none | サーバー名 |
| exclude_databases | [string] | false | none | データベース名を除外するコンマ区切り |

### ExcludeDatabasesPutData

```
{
  "exclude_databases": ["string"]
}
```

### プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| exclude_databases | [string] | false | none | データベース名を除外するコンマ区切り |

### ブラックアウト


```
{
  "id": 0,
  "start_time": "2019-08-24T14:15:22Z",
  "end_time": "2019-08-24T14:15:22Z",
  "object_type": 0,
  "object_id": 0,
  "source": "manual",
  "batch_id": "4da22c97-b7d5-4e31-8c3a-03870ebc7b20",
  "created_by": "string",
  "created_at": "2019-08-24T14:15:22Z"
}
```


### プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| id | integer | false | none | なし |
| start_time | string(date-time) | false | none | なし |
| end_time | string(date-time) ¦ null | false | none | なし |
| object_type | integer | false | none | 100=エージェント、200=サーバー |
| object_id | integer | false | none | なし |
| source | string | false | none | なし |
| batch_id | string(uuid) ¦ null | false | none | なし |
| created_by | string | false | none | なし |
| created_at | string(date-time) | false | none | なし |

### 列挙値

| Property | Value |
|---|---|
| source | manual |
| source | scheduled |
| source | auto |

### BlackoutHistory

```
{
  "id": 0,
  "start_time": "2019-08-24T14:15:22Z",
  "scheduled_end_time": "2019-08-24T14:15:22Z",
  "actual_end_time": "2019-08-24T14:15:22Z",
  "object_type": 0,
  "object_id": 0,
  "source": "manual",
  "batch_id": "4da22c97-b7d5-4e31-8c3a-03870ebc7b20",
  "created_by": "string",
  "created_at": "2019-08-24T14:15:22Z",
  "archived": true
}
```

アーカイブされたブラックアウトレコード。 GET /blackout/?include_history=true ライブ行と混合、および ID が pem.blackout_history でのみ見つかった場合に GET /blackout/{id}によって返されます。

#### プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| id | integer | false | none | なし |
| start_time | string(date-time) | false | none | なし |
| scheduled_end_time | string(date-time) ¦ null | false | none | ブラックアウトが作成されたときの元の end_time |
| actual_end_time | string(date-time) | false | none | 実際に停電が終了したとき |
| object_type | integer | false | none | 100=エージェント、200=サーバー |
| object_id | integer | false | none | なし |
| source | string | false | none | なし |
| batch_id | string(uuid) ¦ null | false | none | なし |
| created_by | string | false | none | なし |

次のページに続く

表 703 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| created_at | string(date-time) | false | none | なし |
| archived | boolean | false | none | 履歴レコードの場合は常に true |

#### 列挙値

| Property | Value |
|---|---|
| source | manual |
| source | scheduled |
| source | auto |

### BlackoutWarning

```
{
  "code": "string",
  "message": "string"
}
```

成功した応答に添付される構造化された非致命的な警告。 `code` は、呼び出し元がディスパッチできる安定した識別子です。 `message` は人間が判読可能です。残りのキーは、警告コンテキストごとにありますObjectId、BlackoutId、ArchivedCount など。

#### プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| code | string | true | none | 安定した警告識別子。既知のコード *manual_post_state_unavailable* COMMIT 後にトグルレスポンス形状の SELECT が失敗した、*in_blackout_post_state_unavailable* 同期フラグの場合、*flag_select_failed* ステータスヘルパーはレコード派生の in_blackout にフォールバック、*history_archive_incomplete* DELETE は予想よりも少ない行をアーカイブしました-RLS /トリガーを調査します pem.blackout_history で)。 |
| message | string | true | none | なし |

### ObjectBlackoutStatus

```
{
  "object_type": 100,
  "object_id": 0,
  "in_blackout": true,
  "manual": {
    "id": 0,
    "start_time": "2019-08-24T14:15:22Z",
    "end_time": "2019-08-24T14:15:22Z",
    "object_type": 0,
    "object_id": 0,
    "source": "manual",
    "batch_id": "4da22c97-b7d5-4e31-8c3a-03870ebc7b20",
    "created_by": "string",
    "created_at": "2019-08-24T14:15:22Z"
  },
  "scheduled": [
    {
      "id": 0,
      "start_time": "2019-08-24T14:15:22Z",
      "end_time": "2019-08-24T14:15:22Z",
      "object_type": 0,
```

(次のページに続く)

(前のページからの続き)


```
      "object_id": 0,
      "source": "manual",
      "batch_id": "4da22c97-b7d5-4e31-8c3a-03870ebc7b20",
      "created_by": "string",
      "created_at": "2019-08-24T14:15:22Z"
    }
  ],
  "auto": {
    "id": 0,
    "start_time": "2019-08-24T14:15:22Z",
    "end_time": "2019-08-24T14:15:22Z",
    "object_type": 0,
    "object_id": 0,
    "source": "manual",
    "batch_id": "4da22c97-b7d5-4e31-8c3a-03870ebc7b20",
    "created_by": "string",
    "created_at": "2019-08-24T14:15:22Z"
  },
  "warnings": [
    {
      "code": "string",
      "message": "string"
    }
  ]
}
```


単一のサーバーまたはエージェントの合成されたブラックアウト状態同期された alert_blackout フラグとソースごとに内訳された完全なレコードセット。

### プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| object_type | integer | false | none | 100 =エージェント、200 =サーバー |
| object_id | integer | false | none | なし |
| in_blackout | boolean | false | none | pem.{server,agent}.alert_blackout の値 — 少なくとも 1 つのソースにこのオブジェクトにアクティブなレコードがある場合に true。 |

次のページに続く

表 706 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| manual | アラートブラックアウト管理 ¦ null | false | none | アクティブな手動レコード常に不定、end_time=NULL または null。 |
| scheduled | *Platform Compatibility* ] | false | none | アクティブなスケジュールされたレコード。 |
| auto | アラートブラックアウト管理 ¦ null | false | none | アクティブな自動レコード到達不能オブジェクトに対してシステムが開始したものまたは null。 |
| warnings | *Platform Compatibility* ] | false | none | オプションの非致命的な警告たとえば、alert_blackout フラグ SELECT が失敗し、in_blackout がレコードリストから派生した場合の flag_select_failed。警告がない場合はフィールドが存在しません。 |

#### 列挙値

| Property | Value |
|---|---|
| object_type | 100 |
| object_type | 200 |

### ObjectBlackoutToggleResult


```
{
  "changed": true,
  "in_blackout": true,
  "manual": {
    "id": 0,
    "start_time": "2019-08-24T14:15:22Z",
    "end_time": "2019-08-24T14:15:22Z",
    "object_type": 0,
    "object_id": 0,
    "source": "manual",
    "batch_id": "4da22c97-b7d5-4e31-8c3a-03870ebc7b20",
    "created_by": "string",
    "created_at": "2019-08-24T14:15:22Z"
```


(次のページに続く)

(前のページからの続き)


```
  },
  "ended_records": [
    {
      "id": 0,
      "source": "manual",
      "scheduled_end": "2019-08-24T14:15:22Z",
      "ended_early": true
    }
  ],
  "warnings": [
    {
      "code": "string",
      "message": "string"
    }
  ]
}
```


オブジェクト中心の PUT 手動有効または DELETE すべて終了の結果。べき等 —2 番目の同一の呼び出しは、同じ後状態で Changed=false を結果ます。

### プロパティ

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| changed | boolean | false | none | 呼び出しが実際に状態を変更した場合 true PUT でレコードが挿入されたか、1 つ以上のレコードが DELETE で終了した場合。 noop 冪等呼び出しの場合は false。 |
| in_blackout | boolean | false | none | alert_blackout の後の状態。 PUT が成功した後、これは常に true です。 DELETE の後は常に false です。 |
| manual | アラートブラックアウト管理 ¦ null | false | none | 現在の手動レコードまたは null。 |

次のページに続く

表 708 – 前のページからの続き

| Name | Type | Required | Restrictions | Description |
|---|---|---|---|---|
| ended_records | [object] | false | none | 終了したレコードDELETE 時にのみ設定されます。PUT 時には常に []。それぞれは、レコード ID、ソース、scheduled_end 文字列 ISO タイムスタンプまたは不定の手動レコードの場合は null、および end_earlyDELETE の時点でレコードの NULL 以外の end_time がまだ将来の場合に true を運びます。 |
| » id | integer | false | none | なし |
| » source | string | false | none | なし |
| » scheduled_end | string(date-time) ¦ null | false | none | なし |
| » ended_early | boolean | false | none | なし |
| warnings | *Platform Compatibility* ] | false | none | オプションの非致命的な警告たとえば manual_post_state_unavailable / in_blackout_post_state_unavailable、COMMIT 後のレスポンス形状の SELECT が失敗した場合。警告がない場合はフィールドが存在しません。 |

### 列挙値

| Property | Value |
|---|---|
| source | manual |
| source | scheduled |
| source | auto |

# 第9章　DEVELOPING

## 9.1 Using the Query tool

PEM には、Postgres サーバーに対してアドホック SQL クエリを発行できる機能が豊富なインタラクティブ開発環境 IDE が含まれています。

クエリツールには、 **ツール> クエリツール**を選択するか、ブラウザツリーの選択したノードのコンテキストメニューからアクセスできます。クエリツールを使用すると、次のことができます。

- アドホック SQL クエリーを発行します。
- 任意の SQL コマンドを実行します。
- SELECT クエリーの結果セットが *[データ出力] タブ* の場合、編集します。
- 現在の接続とトランザクションステータスを構成および表示します。
- 出力パネルに表示されているデータを CSV ファイルに保存します。
- テキスト、グラフィカル、またはテーブル形式< https://explain.depesz.com >と同様に、 SQL ステートメントの実行プランを確認します。
- SQL ステートメントに関する分析情報を表示します。

クエリツールには、2 つのパネルがあります。

- 上部パネルには SQL エディターが表示されます。パネルを使用して、クエリーを入力、編集、または実行できます。また、**History** タブも表示され、セッションで実行されたクエリを表示するために使用できます。編集中にスクラッチパッドを使用して、テキストのスニペットを保持できます。

閉じたスクラッチパッドを再度開くか、新しいスクラッチパッドを開くには、SQL エディターを右クリックし、 **Add Panel** を選択します。同じテクニックを使用して、他のパネルから新しいスクラッチパッドを開くこともできます。

- 下部のパネルには、[ Data Output ] パネルが表示されます。タブ付きパネルには以下が表示されます。
- クエリーによって返された結果セット
- クエリーの実行に関する情報 plandb_tran_5 クエリーの実行に関連するサーバーメッセージ
- サーバーから受信した非同期通知。

### 9.1.1 クエリツールツールバー

ツールバーには、状況依存のアイコンが使用されており、頻繁に実行されるタスクへのショートカットを提供します。

| Icon | Behavior | Shortcut |
|---|---|---|
| Open File | 以前に保存したクエリーを SQL エディターに表示します。 | Accesskey + O |
| Save | クエリーを保存するか、 **Save** メニューにアクセスします。- **Save** を選択して、 SQL エディターパネルで選択したコンテンツをファイルに保存します。 - **Save As** を選択して、新しいブラウザーダイアログボックスを開き、 SQL エディターパネルで選択したコンテンツを保存する新しい場所を指定します。 | Accesskey + S |
| Save Data Changes | データ出力パネルのデータの変更挿入、更新、または削除をサーバーに保存します。 | F6 |
| Find | SQL エディターに表示されるコードを検索、置換、または移動します。- **Find** を選択して検索ターゲットを提供し、 SQL エディターのコンテンツを検索します。 - **Find next** を選択して、次に出現する検索ターゲットを見つけます。 - **以前の検索** を選択して、検索ターゲットの最後に出現した場所に移動します。 - **Pesistent find** を選択して、エディターで検索ターゲットのすべてのオカレンスを識別します。 - **Replace** を選択して、ターゲットの個々のオカレンスを見つけて置換しますプロンプトを表示します。 - **Replace all** を選択して、エディターでターゲットのすべてのオカレンスを見つけて置換します。 - **Jump** を選択して、次の検索ターゲットに移動します。 | Cmd+F Cmd+G Cmd+Shift+G Cmd+Shift+F Alt+G |
| Copy | 表示/編集データモードのときに、[データ出力] パネルで現在強調表示されているコンテンツをコピーします。 | Accesskey + C |
| Paste | 表示/編集データモードで、コピーした行を新しい行に貼り付けます。 | Accesskey + P |
| Delete | 選択した行を削除対象としてマークします。 **Save Data Changes** を選択して、削除したコンテンツを更新します。 | Accesskey + D |

次のページに続く

表 1 – 前のページからの続き

| Icon | Behavior | Shortcut |
|---|---|---|
| Edit | **Edit** メニューのオプションを使用して、テキスト編集ツールにアクセスします。オプションは、クエリツールモードのときに SQL エディタパネルに表示されるテキストに動作します。-現在選択しているテキストをインデントするには、**Indent Selection** を選択します。 - 現在選択しているテキストからインデントを削除するには、 **Unindent Selection** を選択します。 - **Inline Comment Selection** を選択して、SQL スタイルのコメント表記での選択を含む行を囲みます。 - **Inline Uncomment Selection** を選択して、選択した行から SQL スタイルのコメント表記を削除します。 - **Block Comment** を選択して、C スタイルのコメント表記で選択を含むすべての行を囲みます。このオプションはトグルとして機能します。 | Tab Shift+Tab Cmd+/ Cmd+. Shift+Cmd+/ |
| Filter | データの表示/編集モードでデータのフィルタリングと並べ替え基準を設定します。下矢印を選択して、他のフィルタリングと並べ替えオプションにアクセスします。- **Sort/Filter** を選択して、並べ替えとフィルタリングダイアログボックスを開きます。 - 選択したセルの値を含む行のみを表示するには、 **Filter by Selection** を選択します。 - 選択したセルの値を含まない行のみを表示するには、 **Exclude by Selection** を選択します。 - **Remove Sort/Filter** を選択して、以前に選択した並べ替えまたはフィルタリングオプションを削除します。 | Accesskey + F |
| Limit Selector | データセット内の最大行数の値を設定します。 | Accesskey + R |
| Stop | 現在実行中のクエリをキャンセルします。 | Accesskey + Q |
| Execute/Refresh | SQL エディターパネルで選択したクエリーを実行または更新します。下矢印を選択して、他の実行オプションにアクセスします。- トランザクション中にエラーが発生した場合、トランザクションをロールバックするには **Auto-Rollback** を選択します。 - 各トランザクションをコミットするには、 **Auto-Commit** を選択します。トランザクションによって行われた変更は他の人に可視され、クラッシュが発生した場合に持続します。 | F5 |
| Explain | 現在のクエリの説明プランを表示します。`EXPLAIN`の結果は、出力パネルの **Explain** タブにグラフィカルに表示され、 **Data Output** タブにテキスト形式で表示されます。 | F7 |

次のページに続く

表 1 – 前のページからの続き

| Icon | Behavior | Shortcut |
|---|---|---|
| Explain analyze | 現在のクエリーで`EXPLAIN ANALYZE`コマンドを呼び出します。 **Explain Options** メニューから - **Verbose** を選択して、クエリプランに関する詳細情報を表示します。 - **Costs** を選択して、各プランノードの推定起動と合計コスト、および各行の推定行数と推定幅に関する情報を含めます。 - バッファーの使用状況に関する情報を含めるには、 **Buffers** を選択します。 - **Timing** を選択して、起動時間とクエリの各ノードで費やされた時間に関する情報を含めます。 - クエリプランに関する概要情報を含めるには、** Overview** を選択します。 | |
| Commit | トランザクションをコミットします。 | Shift+CTRL+M |
| Rollback | トランザクションをロールバックします。 | Shift+CTRL+R |
| Clear | **Clear** メニューのオプションを使用して、表示内容を消去します。- **Clear Query Window** を選択して、 SQL エディターパネルのコンテンツを消去します。 - ** Clear History** を選択して、**History** タブのコンテンツを消去します。 | Accesskey + L |
| Download as CSV | 現在のクエリの結果セットをコンマ区切りリストにダウンロードします。 **設定-> SQL エディター-> CSV 出力** ダイアログボックスを介して CSV 設定を指定できます。 | F8 |
| Macros | **Manage Macros** を選択して、マクロを作成、編集、またはクリアします。 | |

### 9.1.2 SQL エディターパネル

SQL エディターパネルは、クエリーを手動で提供したり、別のソースからクエリーをコピーしたり、ファイルからクエリーを読み取ることができるワークスペースです。SQL エディターには、構文のカラーリングとオートコンプリート機能があります。

オートコンプリートを使用するには、クエリの入力を開始します。エディターにクエリーの次のオブジェクト名またはコマンドを提案したい場合は、**Control+Space** を押します。たとえば、`\ *SELECT \*FROM\*` 末尾にスペースを入力し、** Control+Space**を押してオートコンプリートオプションのメニューから選択します。

クエリーを入力したら、ツールバーから **Execute/Refresh** を選択します。データベースサーバーは、実行する SQL エディタパネルの完全な内容を受信します。SQL エディターに表示されるコードのセクションのみを実行するには、サーバーに実行するテキストを選択し、**Execute/Refresh** を選択します。

コマンドの実行時にサーバーから返されたメッセージは、 **メッセージ**タブに表示されます。コマンドが成功した場合、[ **Messages** ] タブに実行の詳細が表示されます。

**Edit** メニューは、コードのフォーマットとコメント付けに役立ちます。

- オートインデントを使用して、 **Return** を押して、前の行と同じ深さまでテキストをインデントします。
- 2 行以上を選択して **Tab** を押して、インデントテキストをブロックします。
- コード内の SQL スタイルの実装または削除するか、C スタイルのコメント表記を切り替えます。

ツリーから特定のオブジェクトをドラッグして、長いオブジェクト名の入力に費やす時間を節約することもできます。オブジェクト名を含むテキストは、スキーマ名で完全修飾されます。必要に応じて二重引用符が追加されます。ファンクションとプロシージャーの場合、ファンクション名とパラメーター名がクエリツールに貼り付けられます。

### 9.1.3 [データ出力] パネル

[Data Output] パネルには、最近実行されたクエリによって生成されたデータと統計が表示されます。

#### 9.1.3.1 [データ出力] タブ

[ **Data Output** ] タブには、クエリの結果セットがテーブル形式で表示されます。次のことができます。

- 結果セットから選択してコピーします。
- **Execute/Refresh** オプションを使用して、クエリ実行情報を取得し、クエリ実行オプションを設定します。
- **Download as CSV** を選択して、 **Data Output** タブのコンテンツをコンマ区切りファイルとしてダウンロードします。
- SELECT クエリーの結果セットのデータが更新可能な場合は、編集します。

結果セットは次の場合に更新可能です。

- すべての列は、単一のテーブルから直接選択されるか、実際にはテーブルの列ではありませんたとえば、2 つの列の連結。テーブルから直接選択された列のみを編集できます。他の列は読み取り専用です。
- テーブルのすべての主キー列または OID が結果セットで選択されます。

複数回名前が変更または選択された列も読み取り専用です。

> **注釈**
>
> 更新可能なクエリー結果セットを使用するには、 psycopg2 ドライバーバージョン 2.8 以降をインストールする必要があります。

編集可能および読み取り専用の列は、列ヘッダーの鉛筆と鍵のアイコンを使用して識別されます。

更新可能な結果セットは、データの表示/編集モードのデータグリッドに似ており、同じ方法で変更できます。

自動コミットがオフの場合、データは進行中のトランザクションの一部として作成されます。進行中のトランザクションがない場合、新しいトランザクションが開始されます。トランザクションがコミットされない限り、データの変更はデータベースにコミットされません。

保存中にエラーが発生した場合たとえば、 NOT NULL 制約を使用して列に NULL を保存しようとすると、データの変更がセーブポイントにロールバックされ、進行中のトランザクションで以前に実行されたクエリがロールバックされないことを確認します。

[データ出力] パネルに表示される以前のクエリまたはコマンドからのすべての行セットは、別のクエリを呼び出すと破棄されます。別のクエリツールのブラウザタブを開き、以前の結果を利用可能なままにします。

#### 9.1.3.2 「Explain」タブ

クエリの Explain または Explain Analyze プランを生成するには、ツールバーで **Explain** または **Explain Analyze** を選択します。

メニューから **Explain** および **Explain Analyze** に関連するオプションを選択できます。

> **注釈**
>
> PEM は、JSON 形式で Explain Analyze プランを生成します。

Explain プランが正常に生成されると、 **Explain** パネルの下に 3 つのタブ/パネルが表示されます。

#### 9.1.3.3 グラフィカルタブ

プランを SVG ファイルとしてダウンロードするには、Explain キャンバスの左上隅にある **Download** を選択します。 **SVG としてのダウンロード** は、Internet Explorer ではサポートされていません。

**Explain Analyze** に付随するクエリプランは、 **Data Output** タブで利用できます。

#### 9.1.3.4 [分析] タブ

[ **Anarise** ] タブには、プランの詳細が表形式で表示され、`explain.depsez.com` で使用可能なものと同様の形式が生成されます。テーブルの各行は、Explain Plan ノードのデータを表します。出力には、ノード情報、排他的タイミング、包含タイミング、実際の行対計画的な行、実際の行、計画的な行、またはループを含めることができます。選択した行の子行は、オレンジ色の点でマークされます。

合計クエリ時間の排他的/包括的タイミングのパーセンテージが次の場合

- 90 を超えると、赤が表示されます
- 50 を超えると、オレンジ色が表示されます
- 10 を超えると黄色が表示されます

プランナーが次の理由で行数実際のバース計画を誤って見積もった場合。

- 10 回、黄色が表示されます
- 100 回、オレンジ色が表示されます
- 1000 回、赤が表示されます

#### 9.1.3.5 [統計] タブ

[ **Statistics** ] タブには、2 つのテーブルの情報が表示されます。

- ノードタイプごとの統計は、各ノードタイプが参照された回数を示します。
- テーブルごとの統計は、各テーブルがクエリによって参照された回数を示します。

#### 9.1.3.6 [メッセージ] タブ

**Messages** タブを使用して、最近実行されたクエリに関する情報を表示します。サーバーがエラーを返した場合、エラーメッセージは [ **メッセージ** ] タブに表示され、SQL エディターではエラーの原因となった構文に下線が引かれます。クエリが成功した場合、[ **Messages** ] タブには、クエリが完了するまでに要した時間と取得された行数が表示されます。

#### 9.1.3.7 [通知] タブ

[ **Notifications** ] タブを使用して、PostgreSQL の Listen/Notify 機能を使用して通知を表示します。詳細については、PostgreSQL documentation を参照してください。

例

1. 最初のクエリツールセッションで `LISTEN "foo"` を実行します。

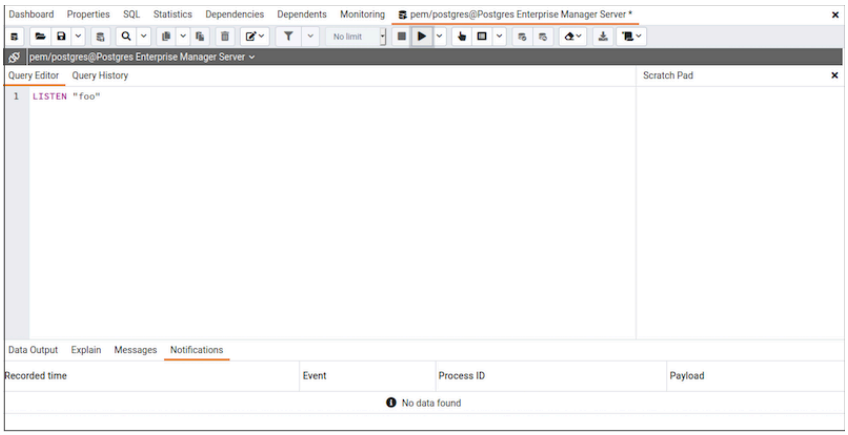



図 1: Query tool - notifications listen

2. 別のクエリツールセッションで、`Notify` コマンドまたは `pg_notify` ファンクションを実行して、ペイロードとともにイベントの通知を送信します。
3. 最初のクエリツールセッションで [ **Notification** ] タブを観察できます。ここには、チャネルの記録時間、イベント、プロセス ID、およびペイロードが表示されます。

### 9.1.4 Query History パネル

**Query History** タブを使用して、現在のセッションのアクティビティを確認します。 [ **Query History** ] タブには、次のような最近のコマンドに関する情報が表示されます。

- クエリーが呼び出された日時。
- クエリのテキスト。

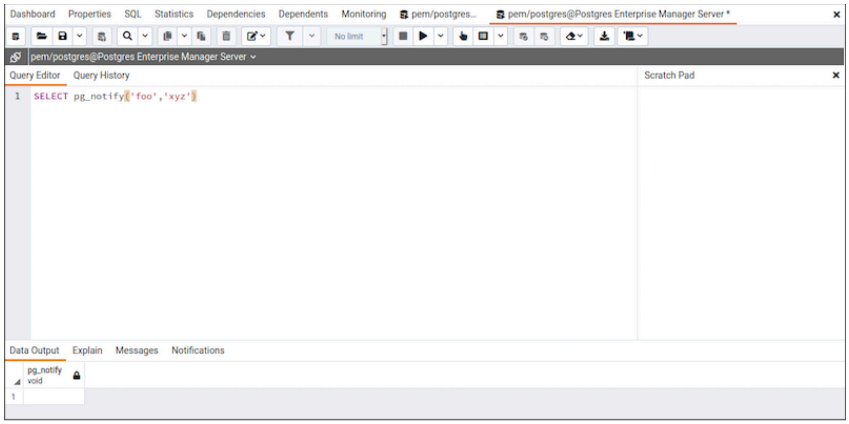



図 2: Query tool - notifications notify

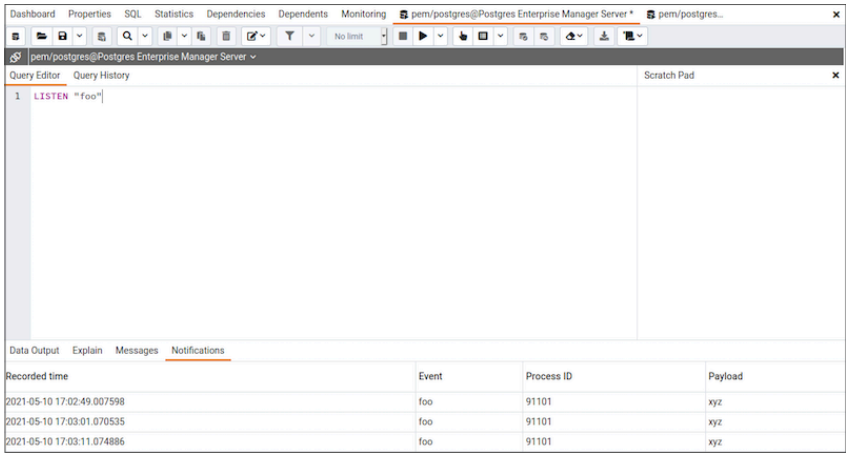



図 3: Query tool - Output Notifications panel

- クエリーによって返された行数。
- サーバーがクエリーを処理し、結果セットを返すまでに要した時間。
- サーバーから返されたメッセージ [ **メッセージ**] タブには記載されていません。
- クエリーのソースツールバーに対応するアイコンで示されます。

pgAdmin によって内部的に生成されたクエリーは、データの表示/編集またはデータの保存操作中に表示または非表示にできます。

**Query History** タブの内容を消去するには、 **Clear > Clearhistory** を選択します。

クエリ履歴は、クエリツールモードで実行している場合、ユーザーごとに各データベースのセッションを超えて維持されます。View/Edit Data モードでは、履歴は保持されません。デフォルトでは、最後の 20 クエリーがデータベースごとに保存されます。`MAX_QUERY_HIST_STORED` 値をオーバーライドすることにより、`config_local.py` でこれを調整できます。

### 9.1.5 接続ステータス

**Connection Status** を使用して、クエリツールでステータスを選択することにより、現在の接続とトランザクションのステータスを表示します。

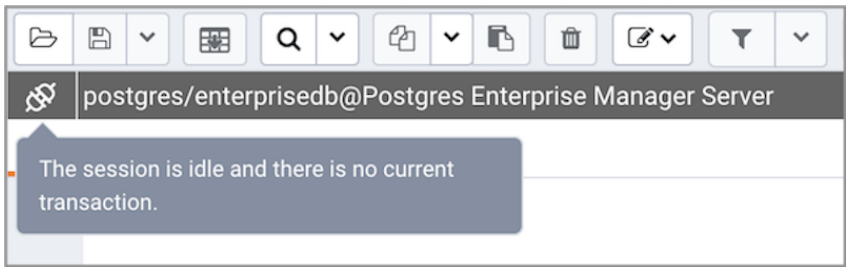



図 4: Query tool - Connection Status button

### 9.1.6 マクロ

クエリツールマクロを使用すると、単一のキーを押すだけで事前定義された SQL クエリーを実行できます。事前定義されたクエリーには、プレースホルダー`$SELECTION$` を含めることができます。マクロが実行されると、プレースホルダーは、クエリツールの [クエリエディター] ペインで現在選択されているテキストに置き換えられます。

マクロを作成するには

1. クエリツールで、 **Macros > Manage Macros** を選択します。
2. 使用するキーを選択します。マクロとクエリの名前を入力します。
3. オプションで、選択プレースホルダーを含めます。
4. **Save** を選択します。

マクロをクリアするには、[ Manage Macros ] ダイアログボックスでマクロを選択し、 **Clear** を選択します。プロンプトに **Yes** と応答します。

すべてのマクロをクリアするには、 **Key** の横にある **Clear** を選択します。プロンプトに **Yes** と応答します。

マクロを実行するには、ショートカットキーを選択するか、 **Macros** メニューから選択します。

## 9.2 Using the Schema Diff tool

Schema Diff は、2 つのデータベーススキーマ間のスキーマオブジェクトを比較します。**Tools** メニューを使用して、Schema Diff にアクセスします。

スキーマ差分を使用すると、次のことができます。

- データベーススキーマソースからターゲットへを比較および同期します。
- データベーススキーマ間の違いを視覚化します。
- ターゲットスキーマオブジェクトの SQL ステートメントの差異をリストします。
- 同期スクリプトを生成します。

選択パネルを開くには、 **Tools > Schema Diff** を選択します。比較するソースおよびターゲットサーバー、データベース、およびスキーマを選択します。オブジェクトを選択したら、[ **Compare** ] を選択します。

> **注釈**
>
> ソースデータベースとターゲットデータベースは、同じメジャーバージョンを使用する必要があります。

スキーマ差分の複数のコピーを個々のタブで開くことができます。新しいブラウザタブでスキーマ差分を開くかどうかを指定するには、[設定] ダイアログボックスを選択します。**新しいブラウザタブで開く** を **true** に設定します。

[ Schema Diff ] パネルは、 Object Comparison パネルと DDL Comparison パネルの 2 つのパネルに分割されています。

### 9.2.1 Schema Diff Object Comparison パネル

[ Object Comparison ] パネルでは、比較する同じメジャーバージョン、データベース、およびスキーマのソースサーバーとターゲットサーバーを選択できます。接続されているか切断されているかにかかわらず、ブラウザツリーの下にリストされているサーバーを選択できます。接続されていないサーバーを選択した場合、そのサーバーを使用する前にパスワードを入力する必要があります。

比較するデータベースを選択します。データベースは同じでも異なっていても、同じまたは別のサーバーからのものでもかまいません。

比較するソーススキーマとターゲットスキーマを選択します。

サーバー、データベース、およびスキーマを選択したら、 **Compare** を選択して比較結果を取得します。

**ファンクション** 、 **マテリアライズドビュー** 、 **テーブル** 、**トリガーファンクション** 、 **プロシージャ** 、および **ビュー s** のリストを使用して、すべての DDL ステートメントを表示しますスキーマオブジェクト。

スキーマオブジェクトをフィルタリングするには、[ Object Comparison ] パネルの右上隅にある **Filter** を選択します。次の基準に従ってスキーマオブジェクトをフィルタリングします。

- **同一** - 同じ SQL ステートメントを使用して両方のスキーマでオブジェクトが見つかった場合、比較結果は同じです。
- **異なる** - 異なる SQL ステートメントを使用して両方のスキーマでオブジェクトが見つかった場合、比較結果は異なります。
- **ソースのみ** - オブジェクトがソーススキーマでのみ見つかり、ターゲットスキーマで見つからない場合、比較結果はソースのみです。
- **ターゲットのみ** - オブジェクトがターゲットスキーマでのみ見つかり、ソーススキーマで見つからない場合、比較結果はターゲットのみです。

Object Comparison パネルでスキーマオブジェクトのいずれかを選択して、そのオブジェクトの DDL ステートメントを DDL Comparison パネルに表示します。

### 9.2.2 Schema Diff DDL Comparison パネル

[DDL Comparison] パネルには、3 つの列が表示されます。

- 最初の列には、ソーススキーマからのオブジェクトの DDL ステートメントが表示されます。
- 2 番目の列には、ターゲットスキーマからのオブジェクトの DDL ステートメントが表示されます。
- 3 番目の列には、ターゲットスキーマオブジェクトの SQL ステートメントの差異が表示されます。

SQL ステートメントの違いを確認するには、すべてのスキーマオブジェクトの DDL ステートメントを確認します。

Schema Diff ツールは、ターゲットスキーマオブジェクトで見つかった差分を使用して SQL スクリプトを生成できます。SQL スクリプトは、ターゲットスキーマオブジェクトをソーススキーマオブジェクトの SQL ステートメントと比較します。スクリプトを生成するには

1. Object Comparison パネルで、スキーマオブジェクトのチェックボックスをオンにします。
2. **Generate Script** を選択します。

新しいタブでクエリツールを開き、クエリエディタで SQL ステートメントの違いを表示するには

1. スキーマオブジェクトを選択します。
2. **Generate Script** を選択します。

スキーマオブジェクトを選択して、[DDL 比較] パネルで生成された差分を確認したが、スキーマオブジェクトのチェックボックスを選択しない場合、PEM は新しいタブでクエリツールを開きます。Query ツールは、 SQL ステートメントの差異をクエリエディターに表示します。

[DDL Comparison] パネルで生成された差分をコピーするには、 **Copy** を選択します。

ターゲットスキーマに SQL ステートメントを適用して、スキーマを同期します。

## 9.3 Using the ERD tool

Entity-Relationship DiagramERD ツールは、データベーステーブル、列、および相互関係のグラフィカル表現を提供するデータベースデザインツールです。ERD は、データベース管理者がデータベースを開発および保守する際に従うための十分な情報を提供できます。ERD ツールにアクセスするには、 **Tools > ERD Tool** を選択します。ERD ツールを使用すると、次のことができます。

- データベーステーブルとその関係をデザインおよび視覚化します。
- 図にノートを追加します。
- テーブルとリンクを揃えて、すっきりした外観にします。
- 図を保存し、後で開いて作業を続行します。
- データベースデザインからすぐに実行できる SQL を生成します。
- 既存のデータベースのデータベース図を生成します。

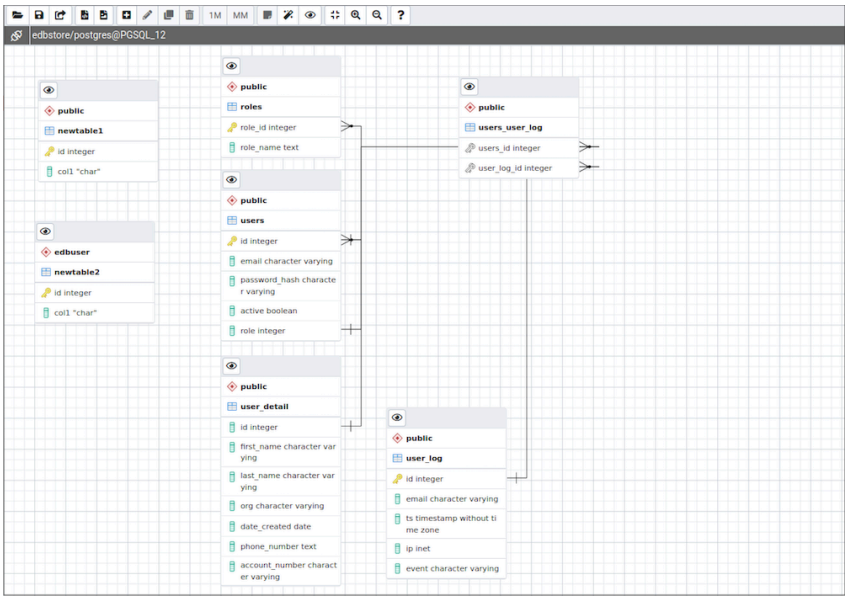



図 5: ERD Tool

ERD ツールの複数のコピーを個々のタブで同時に開くことができます。ERD ツールのコピーを閉じるには、タブバーの右上隅にある **X** を選択します。

### 9.3.1 ツールバー

ERD ツールツールバーは、頻繁に実行されるタスクへのショートカットを提供する状況依存のアイコンを使用します。

### 9.3.2 ファイルオプション

| Icon | Behavior | Shortcut |
|---|---|---|
| Open File | 保存した図をロードします。 | Ctrl + O |
| Save | 保存した図に変更を保存するか、図をファイルに保存します。 | Ctrl + S |
| Save as | 新しいダイアログボックスを開き、図を保存する新しい場所を指定します。 | Ctrl + Shift + S |

### 9.3.3 エクスポートオプション

| Icon | Behavior | Shortcut |
|---|---|---|
| Generate SQL | 図の DDL SQL を生成し、生成された SQL を実行可能な状態でクエリツールを開きます。 | Option + Ctrl + S |
| Download image | ERD 図を画像形式で保存します。 | Option + Ctrl + I |

### 9.3.4 編集オプション

| Icon | Behavior | Shortcut |
|---|---|---|
| Add table | 新しいテーブルを図に追加します。テーブルダイアログボックスに、テーブルの詳細を入力します。 | Option/Alt + Ctrl + A |
| Edit table | 図で選択したテーブルを編集します。テーブルダイアログボックスに、テーブルの詳細を入力します。 | Option/Alt + Ctrl + E |
| Clone table | 完全なテーブル構造のクローンを作成し、自動生成された名前で名前を付けて、図に配置します。 | Option/Alt + Ctrl + C |
| Drop table/link | 選択したテーブルまたはリンクをドロップします。 | Option/Alt + Ctrl + D |

### 9.3.5 テーブルリレーションシップオプション

| Icon | Behavior | Shortcut |
|---|---|---|
| 1M | 1 対多のリレーションシップダイアログボックスを開き、2 つのテーブル間のリレーションシップを追加します。選択したテーブルは参照テーブルであり、リンクの `many`エンドポイントがあります。 | Option/Alt + Ctrl + O |

次のページに続く

表 5 – 前のページからの続き

| Icon | Behavior | Shortcut |
|---|---|---|
| MM | 多対多のリレーションシップダイアログボックスを開き、2 つのテーブル間のリレーションシップを追加します。このオプションは、2 つの関連するテーブルの選択した列からリンクテーブルを作成します。 | Option/Alt + Ctrl + M |

### 9.3.6 ユーティリティオプション

| Icon | Behavior | Shortcut |
|---|---|---|
| Add/Edit note | データベースの設計中に、テーブルノードにメモを作成します。 | Option/Alt + Ctrl + N |
| Auto align | すべてのテーブルとリンクを揃えて、見た目をすっきりさせます。 | Option/Alt + Ctrl + L |
| Show details | 列の詳細を切り替えます。いくつかの列の詳細を表示することを選択することもできます。 | Option/Alt + Shift + D |

### 9.3.7 ズームオプション

| Icon | Behavior | Shortcut |
|---|---|---|
| Zoom to fit | 自動的にズームイン/アウトし、すべてのテーブルをビューに合わせます。 | Option/Alt + Shift + F |
| Zoom in | 図を拡大します。 | Option/Alt + Shift + "+" |
| Zoom out | 図をズームアウトします。 | Option/Alt + Shift + "-" |

### 9.3.8 テーブルダイアログ

テーブルダイアログボックスを使用すると、次のことができます。

- テーブル構造の詳細を変更します。
- 既存のテーブルを編集するか、新しいテーブルを追加します。

### 9.3.9 テーブルノード

テーブルノードには、テーブルの詳細がグラフィカル表現で表示されます。

- トップバー
  - 列の詳細を表示するには、 **詳細トグル** を選択します。
  - ノートを追加すると、 **Note** が有効になります。ノートを変更するには **Note** を選択します。
- 最初の行には、例の `public` のように、テーブルのスキーマ名が表示されます。

- 2 行目には、例の users のように、テーブルの名前が表示されます。
- 残りの行には、テーブルの列名とそのデータ型が表示されます。
  - 例の id integer のように、主キーである列がロックキーとして表示されます。
  - 他の列は、例の email character varying のように、緑色の列として表示されます。
- ノードを選択して、キャンバスにドラッグします。
- テーブルダイアログボックスを開き、テーブルの詳細を変更するには、テーブルノードをダブルクリックするか、ツールバーの **Edit** を選択します。

### 9.3.10 1 対多のリンクダイアログボックス

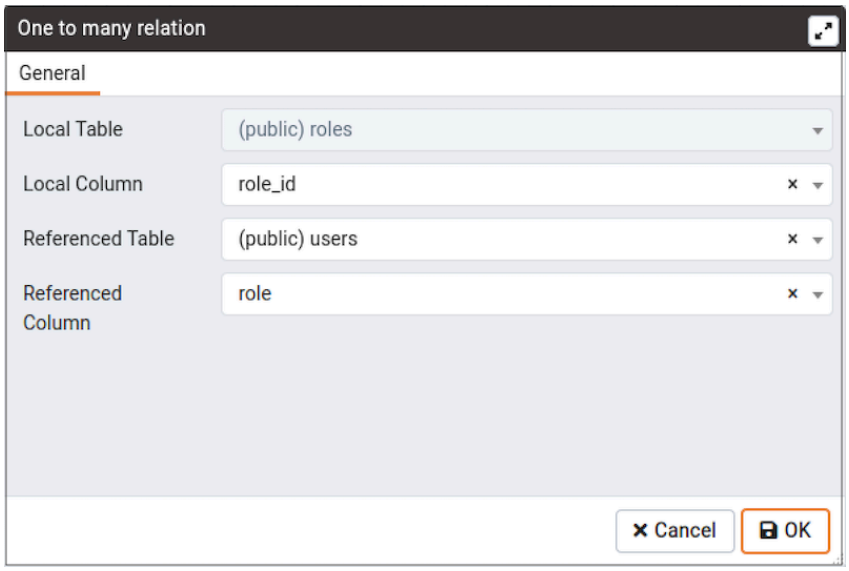



図 6: ERD Tool - One-to-Many Dialog Box

1 対多のリンクダイアログボックスを使用すると、次のことができます。

- 2 つのテーブル間に外部キー関係を追加します。
- 別のテーブルを参照するローカルテーブルを選択します。ローカルテーブルには many エンドポイントがあります。
- 別の列を参照するローカル列を選択します。
- 参照されるテーブルを選択します。参照テーブルには one エンドポイントがあります。
- 参照される列を選択します。

### 9.3.11 多対多のリンクダイアログボックス

多対多のリンクダイアログボックスを使用すると、次のことができます。

- 2 つのテーブル間に多対多のリレーションシップを追加します。
- 2 つのテーブルから派生するリンクされた列を含むリレーションテーブルを作成します。

- リンクされる最初のテーブルである左側のテーブルを選択します。左側のテーブルには、新しいリレーションテーブルとのリンクの one エンドポイントがあります。
- 最初のテーブルの列である左側のテーブル列を選択します。左側のテーブルの列は主キーです。
- リンクされている 2 番目のテーブルである右テーブルを選択します。右テーブルには、新しいリレーションテーブルとのリンクの one エンドポイントがあります。
- 2 番目のテーブルの列である右テーブルの列を選択します。右テーブルの列は主キーです。

### 9.3.12 テーブルリンク

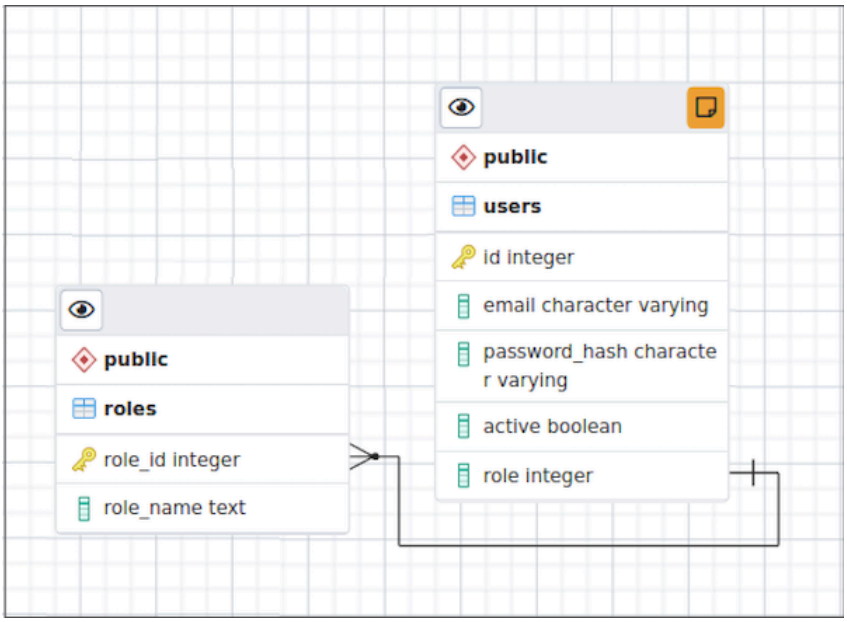



図 7: ERD Tool - Table Link

テーブルリンクは、テーブル間の関係を示します。

- リンクの単線のエンドポイントは、参照である列を示します。
- リンクの 3 本線のエンドポイントは、参照する列を示します。
- 参照または参照している列のいずれかがテーブルから削除されると、リンクがドロップします。
- リンクを選択して、キャンバスにドラッグします。

### 9.3.13 テーブルノート

テーブルにノートを追加するには

1. ツールバーで、 **Add/Edit note** を選択します。
2. ウィンドウで、メモを入力します。

ノートがテーブルに追加されると、テーブルノードで **Note** が有効になります。ノートを確認または更新するには、 **Note** を選択します。

## 9.4 PL debugger

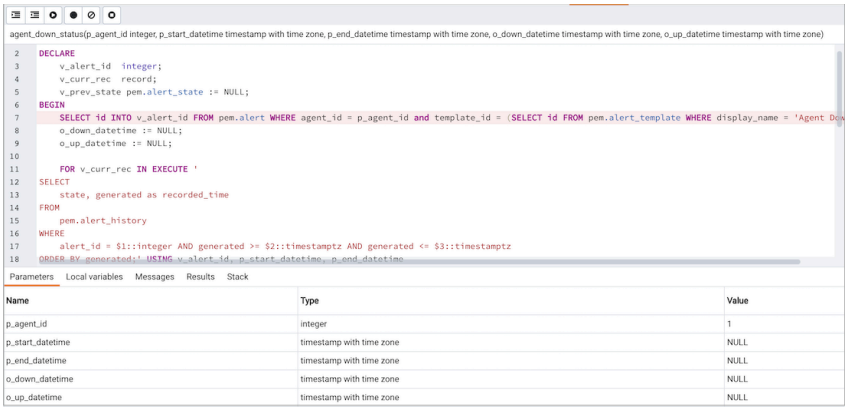



図 8: Debugger page - Parameters tab

デバッガーを使用して、PostgreSQL の PL/pgSQL ファンクション、EDB-SPL ファンクション、ストアドプロシージャ、および EDB Postgres Advanced Server のパッケージをデバッグできます。デバッガーは、PostgreSQL インストールの拡張機能として利用でき、 EDB Postgres Advanced Server の一部として配布されます。デバッガーを使用するにはスーパーユーザー権限が必要です。

デバッガーを使用する前に、`postgresql.conf` ファイルを変更し、サーバーサイドのデバッガーコンポーネントを `shared_preload_libraries` パラメーターの値に追加します。

```
shared_preload_libraries =\$libdir/`other_libraries`/plugin_debugger\
```

`shared_preload_libraries` パラメーターを変更した後、サーバーを再起動して変更を適用します。

デバッガーは、ターゲットファンクションまたはプロシージャのインコンテキストデバッグまたは直接デバッグに使用できます。インコンテキストデバッグにデバッガーを使用する場合、プログラムの最初の行にブレークポイントを設定します。セッションがターゲットを呼び出すと、制御がデバッガーに転送されます。ダイレクトデバッグを使用する場合、デバッガーは、ターゲットに必要なパラメーターの入力を求め、コードをステップ実行できます。

### 9.4.1 インコンテキストデバッグ

プログラムの最初の行にブレークポイントを設定するには、デバッグするオブジェクトの名前を右クリックし、**Debugging > Set Breakpoint** を選択します。デバッガーウィンドウが開き、別のセッションがプログラムを呼び出すのを待機します。

別のセッションがターゲットを呼び出すと、デバッガーにコードが表示され、ブレークポイントを追加したり、行ごとにステップスルーしたりできます。他のセッションは、デバッグが完了するまで一時停止されます。次に、制御はセッションに返されます。

### 9.4.2 ダイレクトデバッグ

直接デバッグにデバッガーを使用するには、ブラウザツリーコントロールで、デバッグするオブジェクトの名前を右クリックし、**Debugging > Debug** を選択します。

[デバッガー] ダイアログボックスのフィールドを使用して、各パラメーターの値を提供します。

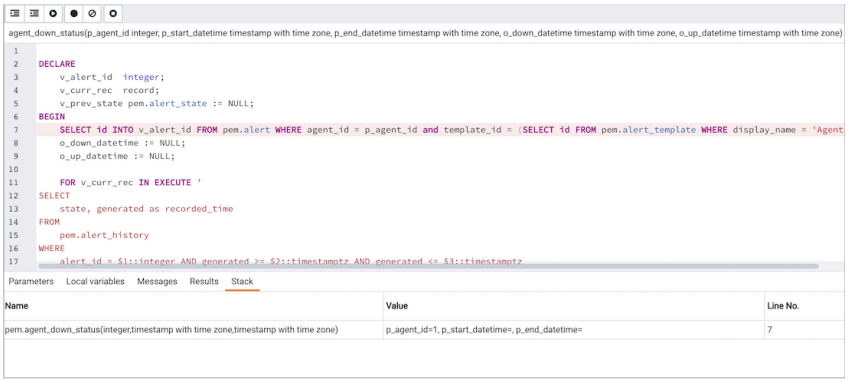



図 9: Debugger page - Stack tab

- **Name** フィールドには、正式なパラメーター名が表示されます。
- **Type** フィールドには、パラメーターのデータ型が含まれます。
- パラメーターが NULL 値の場合、 **Null?** チェックボックスを選択します。
- **Value** ボックスに式が含まれる場合、 **Expression?** チェックボックスをオンにします。
- **Value** ボックスに、プログラムに渡すパラメーター値を入力します。パラメーター値を入力するときに、次のことができます。
  - グリッド上の適切なセルに値を入力します。
  - セルを空のままにして、NULL を表します。
  - 2 つの一重引用符" を入力して、空の文字列を表します。
  - 2 つの一重引用符" だけで構成されるリテラル文字列。
  - PostgreSQL 8.4 以降では、可変個引数ファンクションパラメーターをサポートしています。これらはコンマ区切りの値のリストとして入力でき、必要に応じて引用符またはキャストできます。
- プログラムで **Default Value** ボックスの値を使用する場合、**Usedefault?** チェックボックスを選択します。
- **Default Value** ボックスには、パラメーターのデフォルト値が表示されます。

プログラムに必要な値を入力したら、 **Debug** を選択してプログラムのステップ実行を開始します。

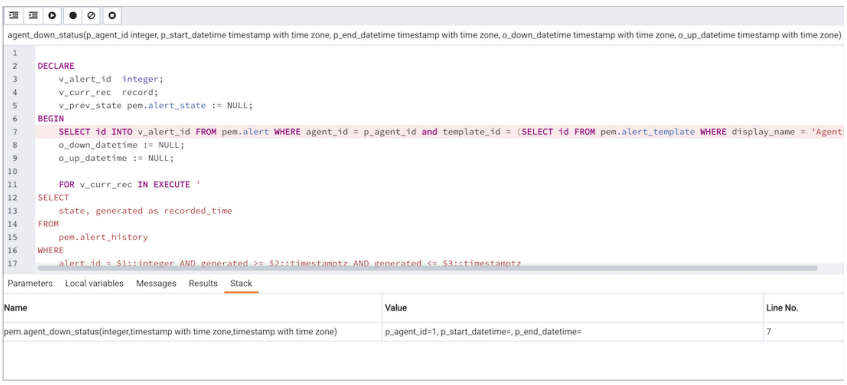



図 10: Debugger page - Stack tab

選択した値は保存され、次回ダイアログボックスを開くときに表示されます。値をクリアするには、**Clear All** を選択します。

### 9.4.3 デバッガーを使用する

メインデバッガーウィンドウは、2 つのパネルと状況依存のツールバーで構成されています。ツールバーアイコンを使用してブレークポイントを管理し、コードにステップインまたはステップスルーします。

| Icon | Action |
|---|---|
| Step into | 現在強調表示されているコード行を実行します。 |
| Step over | コード行を実行し、コードによって呼び出されるサブファンクションをステップオーバーします。サブファンクションは実行されますが、ブレークポイントが含まれていない限りデバッグされません。 |
| Continue/Start | 強調表示されたコードを実行し、プログラムがブレークポイントに遭遇するか完了するまで続行します。 |
| Toggle breakpoint | ブレークポイントを削除せずにブレークポイントを有効または無効にします。 |
| Clear all breakpoints | プログラムからすべてのブレークポイントを削除します。 |
| Stop | プログラムの実行を停止します。 |

デバッガーウィンドウの上部パネルには、プログラム本体が表示されます。行番号の横にある灰色の余白をクリックして、ブレークポイントを追加します。上部パネルの強調表示された行は、これから実行される行です。

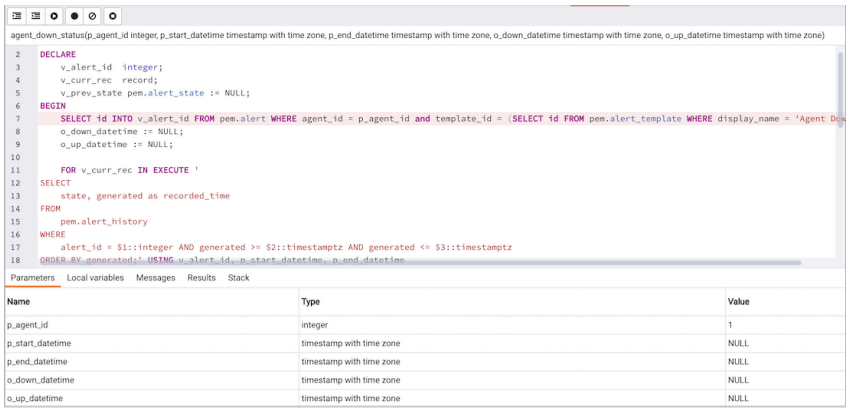



図 11: Debugger main page

デバッガーウィンドウの下部パネルには、プログラムに関する情報を確認できるタブのセットが提供されます。

- **Parameters** タブには、各パラメーターの値が表示されます。
- **ローカル** 変数タブには、プログラム変数の現在値が表示されます。
- **Messages** タブには、サーバーから返されたメッセージが表示されます。
- **Results** タブには、プログラムが完了するとサーバーメッセージが表示されます。
- **Stack** タブには、呼び出されたがまだ完了していないファンクションのリストが表示されます。

プログラムをステップ実行すると、 **ローカル変数**タブに各変数の現在値が表示されます。

図 12: Debugger page - Local variables tab

サブルーチンにステップインすると、 **Stack** タブに、各呼び出し元の名前、各呼び出し元のパラメーター値存在する場合、各呼び出し元内の行番号を含むコールスタックが表示されます。

図 13: Debugger page - Stack tab

呼び出し元を選択してフォーカスをそのスタックフレームに変更し、上部のパネルに呼び出し元の状態を表示します。

プログラムが完了すると、[ **Results** ] タブにサーバーから返されたメッセージが表示されます。プログラムでエラーが発生した場合、[ **Messages** ] タブに詳細が表示されます。

!!!note `ENABLE_DEBUGGER` 構成オプションが false に設定されている場合、デバッガーは無効になっています。!!!