Postgres Enterprise Manager

Release 10.4.1

EnterpriseDB Corporation

Apr 14, 2026

Contents

1 Introduction **3**
1.1 Postgres compatibility . . . 3

2 Postgres Enterprise Manager release notes **5**
2.1 Postgres Enterprise Manager 10.4.1 release notes . . . 5
2.1.1 Security Fixes . . . 5
2.1.2 Bug Fixes . . . 6
2.2 Postgres Enterprise Manager 10.4.0 release notes . . . 6
2.2.1 Highlights . . . 6
2.2.2 Enhancements . . . 6
2.2.3 Changes . . . 7
2.2.4 Bug Fixes . . . 7
2.3 Postgres Enterprise Manager 10.3.1 release notes . . . 8
2.3.1 Changes . . . 9
2.3.2 Bug Fixes . . . 9
2.4 Postgres Enterprise Manager 10.3.0 release notes . . . 9
2.4.1 Highlights . . . 9
2.4.2 Enhancements . . . 10
2.4.3 Changes . . . 11
2.4.4 Bug Fixes . . . 11
2.5 Postgres Enterprise Manager 10.2.0 release notes . . . 14
2.5.1 Highlights . . . 14
2.5.2 Enhancements . . . 14
2.5.3 Changes . . . 15
2.5.4 Bug Fixes . . . 15
2.6 Postgres Enterprise Manager 10.1.2 release notes . . . 17
2.7 Postgres Enterprise Manager 10.1.1 release notes . . . 17
2.7.1 Bug Fixes . . . 17
2.8 Postgres Enterprise Manager 10.1.0 release notes . . . 18
2.8.1 Highlights . . . 18
2.8.2 Enhancements . . . 18
2.8.3 Changes . . . 19
2.8.4 Bug Fixes . . . 19
2.9 Postgres Enterprise Manager 10.0.0 release notes . . . 20
2.9.1 Highlights . . . 20
2.9.2 Enhancements . . . 21

2.9.3 Changes . . . 22
2.9.4 Bug Fixes . . . 22

3 Platform compatibility 23

4 Hardware requirements for installing the PEM server on Linux 25

5 Understanding Postgres Enterprise Manager components and architecture 27
5.1 PEM architecture . . . 28
5.1.1 PEM server . . . 28
5.1.2 PEM agent . . . 29
5.1.3 PEM web application . . . 30
5.1.4 SQL Profiler extension . . . 31

6 Deployment considerations 33
6.1 How PEM licensing works . . . 33
6.1.1 Licensing rules . . . 33
6.1.2 Examples . . . 34
6.2 PEM server and agent connection management mechanism . . . 34
6.2.1 Without PgBouncer . . . 34
6.2.2 With PgBouncer . . . 34
6.3 Authentication options . . . 36
6.3.1 1. Two-factor authentication (2FA) . . . 36
6.3.2 2. Kerberos authentication . . . 36
6.4 Deciding on an SSL model . . . 37
6.4.1 Certificate management models . . . 37
6.5 Securing your deployment . . . 38
6.5.1 Web server security configuration . . . 38
6.5.2 PEM application security configurations . . . 42
6.6 Web server installation options . . . 44
6.6.1 PEM server and web server on separate hosts . . . 44
6.6.2 PEM server and web server on the same host . . . 45
6.7 Sizing PEM deployments . . . 45
6.7.1 Factors affecting resource requirements . . . 45
6.7.2 PEM t-shirt sizing . . . 46
6.8 High availability patterns for PEM deployment . . . 47
6.8.1 Running multiple instances of the PEM backend database . . . 47
6.8.2 Running multiple PEM web application instances . . . 48
6.8.3 Upgrading an HA PEM deployment . . . 48
6.8.4 Load balancers, proxies, and VIPs . . . 48
6.8.5 Reference architectures . . . 49

7 Installing Postgres Enterprise Manager server 55
7.1 Linux x86-64 (amd64) . . . 55
7.1.1 Red Hat Enterprise Linux (RHEL) and derivatives . . . 55
7.1.2 SUSE Linux Enterprise (SLES) . . . 55
7.1.3 Debian and derivatives . . . 55
7.2 Linux Installing Postgres Enterprise Manager server on Linux IBM Power (ppc64le) . . . 56
7.2.1 Red Hat Enterprise Linux (RHEL) and derivatives . . . 56
7.2.2 SUSE Linux Enterprise (SLES) . . . 56
7.3 Linux Installing Postgres Enterprise Manager server on Linux AArch64 (ARM64) . . . 56
7.3.1 Red Hat Enterprise Linux (RHEL) and derivatives . . . 56
7.3.2 Debian and derivatives . . . 56
7.4 Windows . . . 56
7.4.1 Dependencies of the PEM Server and Agent on Linux . . . 56

7.4.2 Installing Postgres Enterprise Manager server on Linux x86 (amd64) . . . 58
7.4.3 Installing Postgres Enterprise Manager server on Linux AArch64 (ARM64) . . . 74
7.4.4 Installing Postgres Enterprise Manager server on Linux IBM Power (ppc64le) . . . 78
7.4.5 Installing Postgres Enterprise Manager server on Windows . . . 83
7.4.6 Creating an EDB repository on an isolated network . . . 91
7.4.7 Configuring the PEM server on Linux . . . 92
7.4.8 Installing Postgres Enterprise Manager in HA Patterns . . . 96

8 Installing Postgres Enterprise Manager agent 131
8.1 Linux x86-64 (amd64) . . . 131
8.1.1 Red Hat Enterprise Linux (RHEL) and derivatives . . . 131
8.1.2 SUSE Linux Enterprise (SLES) . . . 131
8.1.3 Debian and derivatives . . . 131
8.2 Linux Installing Postgres Enterprise Manager server on Linux IBM Power (ppc64le) . . . 132
8.2.1 Red Hat Enterprise Linux (RHEL) and derivatives . . . 132
8.2.2 SUSE Linux Enterprise (SLES) . . . 132
8.3 Linux Installing Postgres Enterprise Manager server on Linux AArch64 (ARM64) . . . 132
8.3.1 Red Hat Enterprise Linux (RHEL) and derivatives . . . 132
8.3.2 Debian and derivatives . . . 132
8.4 Windows . . . 132
8.4.1 Installing Postgres Enterprise Manager agent on Linux x86 (amd64) . . . 132
8.4.2 Installing Postgres Enterprise Manager agent on Linux AArch64 (ARM64) . . . 136
8.4.3 Installing Postgres Enterprise Manager agent on Linux IBM Power (ppc64le) . . . 136
8.4.4 Installing Postgres Enterprise Manager agent on Windows . . . 138

9 Configuring user authentication in PEM 141
9.1 Configuring the PEM server to use two-factor authentication . . . 141
9.1.1 Mail server configuration . . . 142
9.2 Configuring the PEM server to use Kerberos authentication . . . 142
9.2.1 1. Install Kerberos, the PEM server, and the PEM backend database . . . 142
9.2.2 2. Add principals on Kerberos server . . . 143
9.2.3 3. Extract key tables from Kerberos server . . . 143
9.2.4 4. Configure the PEM backend database server . . . 143
9.2.5 5. Obtain and view the initial ticket . . . 144
9.2.6 6. Configure the PEM server . . . 145
9.2.7 7. Browser settings . . . 145
9.3 Configuring the PEM server to use Windows Active Directory domain services for Kerberos authentication (SSPI) . . . 146
9.3.1 1. Install Active Directory, the PEM server, and the PEM backend database server . . . 146
9.3.2 2. Create users in Active Directory to map with service principals . . . 147
9.3.3 3. Extract key tables from Active Directory . . . 149
9.3.4 4. Configure the PEM backend database server . . . 151
9.3.5 5. Obtain and view the initial ticket . . . 151
9.3.6 6. Configure the PEM server . . . 152
9.3.7 7. Browser settings . . . 153

10 Connection pooling using PgBouncer 155
10.1 Preparing the PEM database server . . . 155
10.1.1 Prerequisites . . . 155
10.1.2 Creating users and roles for PgBouncer-PEM connections . . . 156
10.1.3 Updating the configuration files to allow PgBouncer-PEM connections . . . 157
10.1.4 Creating the SSL key and certificate for PgBouncer-PEM authentication . . . 157
10.2 Configuring PgBouncer . . . 158
10.2.1 Prerequisites . . . 159

10.2.2 Configuring PgBouncer . . . 159
10.3 Configuring the PEM agent . . . 161
10.3.1 Prerequisites . . . 161
10.3.2 Configuring a new PEM agent . . . 161
10.3.3 Configuring an existing PEM agent . . . 162

11 Upgrade and migration **165**
11.1 Upgrading a PEM installation . . . 165
11.1.1 Upgrading a PEM installation on a Linux host . . . 165
11.1.2 Upgrading a PEM installation on a Windows host . . . 167
11.1.3 Upgrading an HA PEM installation . . . 169
11.2 Upgrading the PEM backend Postgres database . . . 171
11.3 Adding entries for PEM Agens and admins to connect to PEM server . . . 174
11.4 Adding entries (localhost) for PEM Agens and admins to connect to PEM server . . . 175
11.5 Upgrading SQL Profiler . . . 175
11.6 Upgrading the PEM web server . . . 175
11.6.1 Upgrading the web server on Linux . . . 175
11.6.2 Upgrading PEM-HTTPD on Windows . . . 176
11.7 Moving the PEM server . . . 177
11.7.1 If the PEM agent resides on Linux . . . 182
11.7.2 If the PEM agent resides on Windows . . . 182

12 Splitting an existing PEM server **185**
12.1 Install PEM on the new host . . . 185
12.2 Copy all data from your existing PEM instance . . . 185
12.3 Update the pg_hba.conf file on the new PEM instance . . . 186
12.4 Configure SSL on the new server and reconfigure PEM agents . . . 186
12.4.1 Approach 1 – Reuse certificates from the existing PEM instance . . . 186
12.4.2 Approach 2 – Use different server certificates but the same CA . . . 187
12.4.3 Approach 3 – Replace all server and agent certificates . . . 187
12.4.4 Approach 4 – Use only certificates from an external CA . . . 188
12.5 Disable disconnected agents on each PEM instance . . . 188
12.6 Validate your PEM instances . . . 188

13 Uninstalling **189**
13.1 Uninstalling Postgres Enterprise Manager components on Linux . . . 189
13.1.1 Uninstalling PEM components from RHEL or Rocky Linux or AlmaLinux hosts . . . 189
13.1.2 Uninstalling PEM components from Debian or Ubuntu hosts . . . 190
13.1.3 Uninstalling PEM components from SLES hosts . . . 190
13.2 Uninstalling Postgres Enterprise Manager components on Windows . . . 190

14 Troubleshooting **191**
14.1 Known installation and configuration issues . . . 191
14.1.1 Conflict with existing mod_wsgi . . . 191
14.1.2 Errors from JIT . . . 192
14.1.3 Crash due to bug in RHEL Python 3.9 standard library . . . 192
14.1.4 Blocking TRACK/TRACE Methods in Apache HTTPD . . . 193
14.2 Reconfiguring the PEM server . . . 193
14.3 PEM web application not loading . . . 194
14.4 Error connecting to PostgreSQL server . . . 194
14.5 PEM web server connectivity check . . . 195
14.6 SQL Error - column “ssl” of relation “server” does not exist . . . 195
14.7 SSL Error - Incomplete sslutils upgrade . . . 195
14.8 Nginx SSL_read() failure errors . . . 196

15 Changing the default port **197**
15.1 For RHEL . . . 197
15.1.1 Modify the SELinux configuration . . . 197
15.1.2 Configure the web server (NGINX) . . . 197
15.1.3 Configure the web server (Apache HTTPD) . . . 198
15.2 For Debian and Ubuntu . . . 198
15.2.1 Configure the web server (NGINX) . . . 198
15.2.2 Configure the web server (Apache HTTPD) . . . 199

16 Registering a PEM agent **201**
16.1 How to register PEM agents . . . 201
16.2 Registering a PEM agent using the pemworker utility . . . 201
16.2.1 Linux . . . 201
16.2.2 Windows . . . 202
16.2.3 Authenticating the pemworker utility . . . 203
16.2.4 Enable the PEM Agent service . . . 203
16.2.5 Linux . . . 204
16.2.6 Windows . . . 204
16.3 Verifying PEM agent registration . . . 204
16.3.1 Command syntax . . . 204
16.3.2 Understanding exit codes . . . 205
16.3.3 Usage examples . . . 205
16.4 Unregistering a PEM agent . . . 206
16.5 Advanced usage . . . 206
16.5.1 Using connection strings . . . 206
16.5.2 The registration connection string . . . 206
16.5.3 The agent connection string . . . 207
16.5.4 Setting the agent ID . . . 207
16.5.5 Overriding default configurations - examples . . . 207
16.5.6 Setting the agent ID . . . 208
16.5.7 Providing an existing SSL certificate and key . . . 208
16.5.8 Using a non-root user account to register a PEM agent on Linux . . . 209

17 Registering a Postgres server **211**
17.1 What does registering a Postgres server do? . . . 211
17.2 Using the pemworker utility to register a server . . . 211
17.2.1 Specifying additional parameters for the pemworker connection . . . 213
17.2.2 Examples . . . 214
17.2.3 Using the pemworker utility to unregister a server . . . 214
17.3 Registering a server with automatic server discovery . . . 215
17.4 Manually registering a database server . . . 216
17.5 Verifying the connection and binding . . . 221

18 Managing a PEM server **223**
18.1 Starting and stopping the PEM server and agents . . . 223
18.2 Controlling the PEM server or PEM agent on Linux . . . 224
18.3 Controlling the PEM server or PEM agent on Windows . . . 224
18.4 Controlling the web server on Linux . . . 224
18.4.1 NGINX . . . 225
18.4.2 Apache HTTPD . . . 225
18.5 Controlling the web server on Windows . . . 225
18.6 Managing PEM authentication . . . 225
18.7 Editing the PEM server configuration . . . 226
18.8 Managing security . . . 226

18.8.1 Login roles . . . 226
18.8.2 Group roles . . . 227
18.8.3 Using PEM predefined roles to manage access to PEM functionality . . . 227
18.8.4 Using a team role . . . 228
18.8.5 Object permissions . . . 229
18.9 Server configuration . . . 229
18.10 Server configuration parameters - reference . . . 230

19 Managing Postgres servers **239**
19.1 Starting and stopping monitored database servers . . . 239
19.2 Modifying the pg_hba.conf file . . . 239
19.3 Creating and maintaining databases and objects . . . 241
19.4 Template Linux service script . . . 241

20 Managing a PEM agent **243**
20.1 Managing job notifications . . . 243
20.1.1 Configuring job notifications at job level . . . 243
20.1.2 Configuring job notifications at agent level . . . 244
20.1.3 Configuring job notifications at server level . . . 244
20.2 Managing scheduled jobs . . . 244
20.2.1 Provide general information about the job . . . 244
20.2.2 Define the job steps . . . 244
20.2.3 Define the job schedule . . . 245
20.2.4 Configure the email notifications . . . 246
20.3 Modifying agent configuration . . . 246
20.3.1 Contents of the configuration file . . . 246
20.3.2 Contents of the registry . . . 249
20.3.3 About alert threads . . . 251
20.4 Reviewing or modifying agent properties . . . 252
20.5 Setting agent privileges . . . 252
20.5.1 Root user versus non-root user . . . 252
20.5.2 Functionality affected by limiting operating system privileges . . . 253
20.5.3 Functionality affected by limiting database privileges . . . 254
20.5.4 Error handling . . . 255

21 Managing SSL certificates **257**
21.1 Web server certificates . . . 257
21.1.1 Replace the web server certificates (NGINX) . . . 257
21.1.2 Replace the web server certificates (Apache HTTPD) . . . 258
21.2 PEM backend database server and agent connection certificates . . . 258
21.2.1 Certificates and key files on the PEM server . . . 258
21.2.2 Certificates and key files for PEM agents . . . 259
21.2.3 PEM certificate renewal . . . 259
21.3 Generate a common agent certificate and key pair . . . 259
21.4 Use certificates and keys signed by trusted CA . . . 261
21.4.1 Replace the server SSL certificates with certificates signed by a trusted CA . . . 261
21.4.2 Replace the agent SSL certificates with certificates signed by a trusted CA . . . 262
21.5 Testing certificates . . . 263
21.5.1 Validate a certificate against the root certificate . . . 263
21.5.2 Make a test connection to the PEM backend database . . . 263

22 Managing configuration settings **269**

23 Troubleshooting agent issues **271**
23.1 Restoring a deleted PEM agent . . . 271

23.2 Updating configuration file path (agent status displaying as unknown) . . . 271
23.3 Using the command line to delete a PEM agent with down or unknown status . . . 272

24 Accessing the web interface 273
24.1 PEM client object browser . . . 275
24.2 PEM tabbed browser window . . . 276
24.3 Using chart, graph, and table controls . . . 278
24.3.1 Personalizing a graphic . . . 278
24.4 Browser toolbar . . . 279
24.5 PEM menu bar . . . 279
24.5.1 File menu . . . 279
24.5.2 Object menu . . . 279
24.5.3 Management menu . . . 280
24.5.4 Dashboards menu . . . 280
24.5.5 Tools menu . . . 281
24.5.6 Help menu . . . 281
24.6 Controlling and customizing charts, graphs, and tables . . . 282

25 Connecting and disconnecting a database server 283
25.1 Disconnecting from a server . . . 283

26 Monitoring performance 285
26.1 Probes . . . 287
26.1.1 System probes . . . 287
26.1.2 Customizing probes . . . 287
26.2 Alerts . . . 295
26.2.1 Viewing the alerts via Global dashboard . . . 295
26.2.2 Viewing the alerts via Alerts dashboard . . . 296
26.2.3 Customizing the Alerts dashboard . . . 296
26.3 Managing monitoring settings using profiles . . . 316
26.3.1 Purpose of using profiles . . . 316
26.3.2 Key concepts . . . 317
26.3.3 Accessing Manage Profiles . . . 317
26.4 Notifications . . . 320
26.4.1 SMTP . . . 320
26.4.2 Webhooks . . . 321
26.4.3 SNMP . . . 327
26.4.4 Nagios . . . 327
26.5 Capacity Manager . . . 329
26.5.1 Capacity Manager templates . . . 331
26.5.2 Capacity Manager metrics - reference . . . 331
26.6 Audit Manager . . . 334
26.6.1 Setting the EDB Postgres Advanced Server instance service ID . . . 335
26.6.2 Setting the EDB Audit Configuration probe . . . 335
26.6.3 Configuring audit logging with the Audit Manager . . . 335
26.6.4 Viewing the log with the Audit Log dashboard . . . 337
26.7 Log Manager . . . 338
26.7.1 Reviewing the Server Log Analysis dashboard . . . 342
26.8 Charts . . . 343
26.8.1 Creating a custom chart . . . 343
26.8.2 Importing a Capacity Manager template . . . 345
26.9 Dashboards . . . 345
26.9.1 Dashboards overview . . . 345
26.9.2 Available dashboards . . . 347

26.9.3 Dashboard configuration . . . 348
26.9.4 Custom dashboards . . . 349
26.10 Remote monitoring . . . 349

27 Monitoring Barman **351**
27.1 Prerequisites for monitoring Barman . . . 351
27.2 Configuring a Barman server . . . 351
27.2.1 Using PEM web client . . . 351
27.2.2 Configure . . . 351
27.2.3 Editing . . . 352
27.2.4 Using pemworker command line . . . 352
27.3 Viewing the Barman server details on a PEM dashboard . . . 353

28 Monitoring EDB Postgres Distributed **355**

29 Monitoring Failover Manager **357**
29.1 Replacing a primary node . . . 358
29.2 Switchover EFM cluster . . . 358

30 Monitoring Replication Server **359**

31 Monitoring event history **361**

32 Tuning performance **363**
32.1 Performance Diagnostic . . . 363
32.1.1 Prerequisites . . . 363
32.1.2 Using the Performance Diagnostic dashboard . . . 364

33 Profiling workloads **369**
33.1 Installing the SQL Profiler extension . . . 369
33.1.1 Installing the package . . . 369
33.1.2 Enabling the extension . . . 370
33.2 Using SQL Profiler . . . 371
33.2.1 Permissions for SQL profiler . . . 371
33.2.2 Creating a trace . . . 371
33.2.3 Opening an existing trace . . . 372
33.2.4 Filtering a trace . . . 372
33.2.5 Deleting a trace . . . 373
33.2.6 Viewing scheduled traces . . . 373

34 Reports **375**
34.1 Custom report templates . . . 375
34.2 Viewing system reports . . . 375
34.2.1 Custom report templates . . . 376
34.2.2 Viewing system reports . . . 381

35 PEM command line interface **385**
35.1 Example . . . 388

36 EDB Postgres Enterprise Manager REST APIs v1.0.0 **389**
36.1 Token . . . 389
36.1.1 TokenGeneration . . . 389
36.1.2 Parameters . . . 391
36.1.3 Responses . . . 391
36.1.4 Response Schema . . . 391
36.1.5 Response Headers . . . 392

36.1.6 delete_*token* . . . 392
36.1.7 Parameters . . . 393
36.1.8 Responses . . . 393
36.1.9 Response Schema . . . 393

37 Using the Query tool **1011**
37.1 The Query tool toolbar . . . 1012
37.2 The SQL editor panel . . . 1014
37.3 The Data Output panel . . . 1014
37.3.1 Data Output tab . . . 1014
37.3.2 Explain tab . . . 1015
37.3.3 Graphical tab . . . 1015
37.3.4 Analysis tab . . . 1015
37.3.5 Statistics tab . . . 1015
37.3.6 Messages tab . . . 1016
37.3.7 Notifications tab . . . 1016
37.4 Query History panel . . . 1017
37.5 Connection status . . . 1017
37.6 Macros . . . 1017

38 Using the Schema Diff tool **1019**
38.1 Schema Diff Object Comparison panel . . . 1019
38.2 Schema Diff DDL Comparison panel . . . 1020

39 Using the ERD tool **1021**
39.1 Toolbar . . . 1021
39.2 File options . . . 1021
39.3 Export options . . . 1022
39.4 Editing options . . . 1022
39.5 Table relationship options . . . 1022
39.6 Utility Options . . . 1022
39.7 Zoom options . . . 1023
39.8 Table dialog . . . 1023
39.9 Table node . . . 1023
39.10 One-to-many link dialog box . . . 1023
39.11 Many-to-many link dialog box . . . 1025
39.12 Table link . . . 1025
39.13 Table notes . . . 1026

40 PL debugger **1027**
40.1 In-context debugging . . . 1028
40.2 Direct debugging . . . 1028
40.3 Using the debugger . . . 1029

Welcome to Postgres Enterprise Manager (PEM). PEM consists of components that provide the management and analytical functionality for your EDB Postgres Advanced Server or PostgreSQL database. PEM is based on the Open Source pgAdmin 4 project.

CHAPTER 1

Introduction

PEM is a comprehensive database design and management system. PEM is designed to meet the needs of both novice and experienced Postgres users alike, providing a powerful graphical interface that simplifies the creation, maintenance, and use of database objects.

1.1 Postgres compatibility

Supported versions of Postgres for PEM 10.x:

	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

CHAPTER 2

Postgres Enterprise Manager release notes

The Postgres Enterprise Manager documentation describes the latest version of Postgres Enterprise Manager 10.

Postgres Enterprise Manager version	Release Date
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 10.4.1 release notes

Released: 09 April 2026

New features, enhancements, bug fixes, and other changes in Postgres Enterprise Manager 10.4.1 include the following:

2.1.1 Security Fixes

Description	Addresses
Updated Cryptography to 46.0.6 to fix CVE-2026-34073.	
Updated requests to 2.33.1 to fix CVE-2026-25645.	
Updated Pygments to 2.20.0 to fix CVE-2026-4539.	
Updated ajv to 8.18.0 to fix CVE-2025-69873.	
Updated Lodash to 4.18.1 to fix CVE-2026-4800 and CVE-2026-2950.	

2.1.2 Bug Fixes

Description	Addresses
Fixed issue of selecting parent node selects all its children nodes.	
Fixed scroll issue in Audit/Probe/Server log dashboards.	
Fixed the issue where user with pem_admin role is able to view all servers irrespective of team access.	

2.2 Postgres Enterprise Manager 10.4.0 release notes

Released: 20 March 2026

New features, enhancements, bug fixes, and other changes in Postgres Enterprise Manager 10.4 include the following:

2.2.1 Highlights

- New user-defined reports which can run on any monitored server or the PEM server.
- New `configure-pem-server.sh` options that enable much more flexible installations.
- Fully revised documentation for HA PEM deployments using new configuration options.
- Support for SLES 15 SP7.

2.2.2 Enhancements

Description	Addresses
Introduced a comprehensive custom reports feature allowing users to create, execute, and manage user-defined report templates on PEM servers or monitored servers, with support for report execution as scheduled jobs and integration with the PEM API.	
Introduced new options for *configure-pem-server.sh* to enable more flexible PEM deployments. This includes the ability to provide passthrough options for the host agent, and to run the script on primary and standby nodes of a running HA Postgres cluster.	
Implemented profile reassignment feature that allows you to move alert profiles from one target to another during the deletion process. This overrides the default behavior of simply removing the profile, ensuring your monitoring configurations remain intact even when a specific profile is deleted.	
Implemented connection string support for pemWorker agent connections, simplifying deployment by allowing users to specify all connection parameters in a single string and reducing the need for separate SSL certificate configuration.	
<details><summary>Added provisional support for OAuth 2.0 Client Credentials (M2M) authentication for webhook endpoints.</summary> You can now configure webhooks to use OAuth: first, fetching a token, then using that token to authenticate subsequent HTTP requests. The support is marked as provisional, meaning it is suitable for production use, but future minor releases may make breaking changes. </details>	
Added a new *verify-registration* command for pemworker that allows users to verify if the agent is properly registered with the PEM server, enabling independent validation of agent registration status without inspecting log files.	
Added a scheduled system job that automatically cleans up old event history records, helping users manage database growth and maintain optimal performance over time.	

continues on next page

Table 4 – continued from previous page

Description	Addresses
Introduced a comprehensive REST API v16 for profile management, enabling developers to programmatically create, retrieve, update, and manage alert profiles through standard API endpoints.	
Introduced new privilege roles: *pem_list_barman* to list the BARMAN server, *pem_list_all_assets* that can see all monitored resources/assets irrespective of the *team* field, *pem_manage_assets* to add/remove a database server, an agent or a tool (e.g. BARMAN), and *pem_assign_profile* to assign a profile to an agent/server.	
Added support to disable alert reminder emails, giving users fine-grained control over their notification preferences to reduce email fatigue from repeated alert notifications.	
Extended email address validation to support addresses beginning with a "+" symbol and longer domain names, accommodating users with modern email addressing schemes like Gmail's plus-addressing feature.	
Added support for SUSE Linux Enterprise Server 15 Service Pack 7 (SLES/15/SP7).	

2.2.3 Changes

Description	Addresses
The PEM web application now uses the edb-python 3.12 interpreter on all Linux platforms except Ubuntu 24.04.	
PEM packages for RHEL 9 are now on RHEL 9.6 so that the package can work on both RHEL 9.6 and 9.7. Consequently, users on PEM 10.3 (ARM) and PEM 10.3.1 (Intel/ARM) must upgrade to PEM 10.4 to maintain full support.	
Merged pgAdmin 9.10 codebase into PEM, bringing the latest pgAdmin features, bug fixes, and improvements to the PEM user interface and core functionality.	

2.2.4 Bug Fixes

Description	Addresses
Fixed an issue in the copy alert feature where the Cleared Alerts flag was not getting copied, ensuring all alert properties are properly duplicated when copying alerts.	41904
Fixed the qtliteral function to properly handle SQL formatting issues by delegating escape handling to the psycopg library, eliminating manual quote doubling bugs.	55011, 55481
Migrated *log_connections* values from 'true'/'false' to standard PostgreSQL format 'on'/'off' in *log_configuration* tables, ensuring consistency with PostgreSQL configuration standards.	55651
Fixed the issue where the system configuration report was throwing an error due to missing tablespace directory. Also created an alert template 'Tablespace Directory Missing' to proactively notify users about such ghost tablespaces.	53777
Fixed unnecessary certificate path checks and creation when agent SSL certificates and keys are specified via *--use-agent-sslcrt* and *--use-agent-sslkey*, streamlining agent registration with pre-existing certificates.	56476
Fixed an issue where 'alerting since' displays incorrect future time when database timezone is in non-UTC timezone.	57785
Fixed login error when using PG/EPAS as the database of user settings via *CONFIG_DATABASE_URI*.	57193

continues on next page

Table 6 – continued from previous page

Description	Addresses
Updated urllib3 to version 2.6.3 to fix CVE-2026-21441, protecting users from potential security vulnerabilities in HTTP client operations.	
Bumped require.js version from 2.3.6 to 2.3.8, incorporating upstream bug fixes and security improvements to the JavaScript module loader.	
Fixed an issue where duplicate chart names with the same target level were incorrectly allowed, ensuring chart uniqueness and preventing user confusion when managing dashboards.	
Fixed a problem where resetting email templates returned stale data instead of reflecting the actual reset values, ensuring users see accurate template content after performing a reset.	
Fixed the Transaction ID Exhaustion (Wraparound) probe to cast values to Float instead of integer, preventing data overflow and providing accurate percentage calculations for transaction ID usage monitoring.	
Resolved profile drafting logic issues affecting event history logging and UI notifications, ensuring profile changes are accurately tracked and users receive proper feedback during profile management.	
Added auto_created column to profile_alert_configs table and updated deletion logic to prevent accidental removal of system-created alerts from profiles, protecting critical monitoring configurations.	
Improved the monitored server connection assignment in pemworker using modern C++ constructs with boost::condition_variable, enhancing thread safety and reducing CPU usage during connection allocation.	
Fixed the issue where agent registration options *-o log_level=xxx* and *-o log_location=yyy* were ignored during registration, ensuring users can properly customize logging configuration during agent setup.	
Modified the server PUT API to enforce proper role-based restrictions, preventing users without the databaseServerRegistration role from editing server properties and enhancing security by encrypting sensitive connection parameters.	
Fixed SSL errors in server registration and refactored the server-registration logic to use schema-wise mapping of server insert statements, improving code maintainability and registration reliability.	
Optimized the *trg_server_info_flag_mv* trigger to only refresh materialized views when the *server_version_id* column is updated instead of on all updates, significantly reducing unnecessary database overhead.	
Added a trigger to update the is_leader flag in the pem_host_and_server table on the basis of pemdata.server_info table.	

2.3 Postgres Enterprise Manager 10.3.1 release notes

Released: 25 February 2026

!!!Important PEM Agent 10.3.1 Compatibility and Registration The `pemworker` utility in PEM agent 10.3.1 can't register a new server if the PEM server version is earlier than 10.3. This is a known issue scheduled for resolution in a future release.

Recommended Workarounds _ If you need to register a server with a PEM server version earlier than 10.3, you can use one of the following methods:

1. **Registration via Web UI** _ Register the server using the PEM web application interface.
2. **Staged agent upgrade** _ First register the server using PEM agent version 10.2, then upgrade the agent to version 10.3.1.

2.3.1 Changes

Description	Addresses
PEM now depends on the OS-provided libcurl package on RHEL 8 x86 (this was already the case for all other platforms). The deprecated <code>libcurl-pem</code> package can be safely removed after upgrade.	

2.3.2 Bug Fixes

Description	Addresses
Cryptography was updated to 46.0.5 (Fixes: CVE-2026-26007).	
urllib3 was updated to 2.6.3 (Fixes: CVE-2025-66418, CVE-2025-66471, CVE-2026-21441).	
Authlib was updated to 1.6.7 (Fixes: CVE-2025-68158).	
Pillow was updated to 12.1.1 (Fixes: CVE-2026-25990, for the platforms using Python 3.10+).	
Werkzeug was updated to 3.1.5 (Fixes: CVE-2025-66221, CVE-2026-21860).	
PyNaCl was updated to 1.6.2 (Fixes: CVE-2026-26007).	
pyasn1 was updated to 0.6.2 (Fixes: CVE-2026-23490).	
RequireJS was updated to 2.3.8 (Fixes: CVE-2024-38999).	
Swagger-UI was updated to 5.31.0 (Fixes: CVE-2021-46708, CVE-2018-25031).	
Axios was updated to 1.13.5 (Fixes: CVE-2025-27152, CVE-2026-25639, CVE-2025-58754, CVE-2024-57965).	
Plain SQL restore now runs with the 'restrict' option to prevent harmful psql meta-commands (Fixes CVE-2025-13780).	
PEM now masks the secret key for the 'restrict' option in the process watcher when restoring plain SQL files (Fixes CVE-2026-1707).	

2.4 Postgres Enterprise Manager 10.3.0 release notes

Released: 25 November 2025

!!!Important When upgrading to PEM 10.3 on a RHEL 9 system where the `libcurl-minimal` package is installed, you must pass the `dnf --allow-erasing` option.

New features, enhancements, bug fixes, and other changes in Postgres Enterprise Manager 10.3 include the following:

2.4.1 Highlights

- New **Profiles** feature to ease the management of probe and alert configuration across many server and agents
- New options to make it easier to bring your own SSL infrastructure with PEM
- New built-in alerts for EFM cluster health
- FIPS-compliant cryptography module
- Support for Postgres 18

2.4.2 Enhancements

Description	Addresses
Added support for PostgreSQL, EDB Postgres Extended Server and EDB Postgres Advanced Server. These Postgres versions can be monitored, or used as PEM backend servers.	
Implemented a comprehensive Profile Management feature for PEM that enables administrators to create, manage, and assign profiles containing predefined probe and alert configurations to multiple agents and servers. The new Profile Management feature allows users to create, manage, and assign profiles with predefined probe and alert configurations to multiple agents and servers, streamlining monitoring setup.	
New granular Postgres roles have been introduced to control user access to various pgAdmin-based tools, including the Query Tool and Schema Diff Tool. The following new roles have been added: - query_tool - schema_diff_tool - debugger_tool - erd_tool - grant_wizard - backup_tool - restore_tool - import_export_data_tool - search_objects_tool - maintenance_tool	
Implemented a FIPS-compliant SHA256-based PEM cryptography module for password storage. Implemented a new cryptographic module using SHA256 for encryption of stored passwords, deprecating the old SHA1-based crypto. New installations of PEM 10.3 will use the new module only. Systems that are upgraded to 10.3 will use the new module to encrypt passwords, and the old module to decrypt existing ones. A future release of PEM will provide a mechanism to migrate existing passwords to the new module.	
Added an alert template 'EFM Fewer Than N Nodes Active'. A new alert template, 'EFM Fewer Than N Nodes Active', has been added to monitor the number of active nodes in an EFM cluster. The new alert template is not set to auto-create, so users should create alerts from them as required. The serverreplication solutionmust be set to EFM to use this alert.	
Introduced a new API version v15 which supports *profile_id* assignment in the server and agent endpoints.	
Added new options to suppress creation of SSL certificates during PEM configuration and agent registration. If you are installing PEM in an environment where SSL certificates are provided externally, you can suppress generation of SSL certificates by PEM using these new options when invokingconfigure_pem_server.sh. The--skip-db-certificateoption has the following effects: - Prevents the creation of a server certification for Postgres. - Disables the check for thesslutilsextension and the creation of a CA certificate - and Sets-o check_server_certificate_expiry=falseand-o skip_db_certificate=trueduring local agent registration meaning no client certificates will be generated and the certificate renewal system job will be disabled. The--skip-web-certificateoption prevents the creation of a self-signed web server certificate. The--use-trusted-ssl-db-certificatesoption is a convenience flag which is equivalent to both--skip-db-certificateand--no-hba-change.	
Added Tab preservation for PEM-related Panels. Tab preservation has been implemented for PEM-related panels, ensuring that your open tabs remain active when navigating or restarting the application.	
Implemented Event History Tracking for Manage Profile. Event History Tracking has been implemented for the Manage Profile feature to track all profile-related changes for audit purposes.	
Added role for managing profiles. A new role,pem_manage_profile, has been added for profile management. This role inherits permissions frompem_probe_configandpem_alert_config.	

continues on next page

Table 9 – continued from previous page

Description	Addresses
PEM will now automatically blackout any servers or agents that have not been reachable for 48 hours. A scheduled job runs hourly to check the server/agent heartbeat against theserver_contact_timeoutconfiguration. If unreachable for 48 hours, the object is blacked out; if reachable again, the blackout is undone.	
Added built-in alert templates for "EFM Agent Down" and "EFM Missing Primary". These new alert templates are not set to auto-create, so users should create alerts from them as required. The serverreplication solutionmust be set to EFM to use these alerts.	
The PEM installer for Windows now integrates sslutils 1.4, which adds support for specifying a certificate expiration period through parameters.	
Upgraded Boost package to 1.89 for PEM agent Windows installer.	
Upgraded Postgresql server package to 17.6-2 in PEM server Windows installer.	
Added contextual logging for jobs. Contextual logging has been added for PEM agent jobs to provide more detailed and relevant information for troubleshooting inworker.log.	

2.4.3 Changes

Description	Addresses
Using variable PEM_HOST instead of HOST not to conflict with the environment variable. TheHOSTvariable used for the PEM DB server host address during PEM installation in web mode has been changed toPEM_HOSTto avoid conflicts with existing environment variables.	
PEM will no longer create anagent_<id>user when registering an agent with--pem-agent-userspecified. In previous versions, PEM would create a Postgres user for every registered agent, even if an existing user was specified with--pem-agent-user. This is no longer the case in PEM 10.3. A Postgres user will only be created if it is not specified.	43623
The Performance Diagnostic selection bar is now set to full width by default. In Performance Diagnostic, the purple selection bar now defaults to full width to ensure the query list is populated immediately, improving usability.	
PEM now enables Barman probes by default. In previous versions, the Barman probes were disabled by default and had to be enabled before any data would be collected from registered by any instances. Since the Barman probes do nothing when there are no Barman instances, and are essential for all Barman functionality, we now enable them by default when PEM is configured.	
The SNMP++ library has been upgraded to version 3.6.3 from the previous 3.6.0.	
The 'System Wait' and 'Session Wait' dashboard options are hidden when monitoring PostgreSQL and EDB Postgres Advanced Server (EPAS) versions 18 and above.	
Upgraded MUI v5 to v7.	

2.4.4 Bug Fixes

Description	Addresses
Fixed an issue where theconfigure-pem-server.shscript would overwrite thepg_hba.conffile, even when--no-hba-changewas specified. Resolved an issue where theconfigure-pem-server.shscript would overwrite thepg_hba.conffile, even when--no-hba-changewas specified, due to an overly strict regular expression for checking existing HBA entries.	54064
Fixed an issue whereby the PEM agent was not able to send email iflibcurl-minimalis installed on RHEL 9. The PEM Agent requires thelibcurlpackage in order to send SMTP to a mail server. We found that the package-level requirement forlibcurlwas met by a packagelibcurl-minimal, frequently found in RHEL 9 builds. However this package does not provide the libraries needed for SMTP, so PEM 10.3 now depends onlibcurl-full. When installing PEM 10.3 packages on a system withlibcurl-fullyou must pass the--allow-erasingoption to allow dnf to removelibcurl-minimal.	
Fixed failover detection logic for Patroni and EFM clusters by replacing server ID-based filtering with cluster name-based filtering and adding a 30-second time window around demotion events to find matching promotions.	52030
Added PEM configuration flag--asb-host-nameto specify the ASB host address of monitored server A new PEM configuration flag,--asb-host-name, has been added to explicitly specify the host address for the monitored server's ASB, resolving an issue where multi-host strings were incorrectly used for the ASB host.	
Fixed an issue where downloading charts would only produce a JPEG file, regardless of the user's selected download format.	
Resolved an error that occurred when attempting to download the complete trace from the SQL Profiler.	
Fixed UI validations for alerts and probes. Added missing UI validation in Manage Alerts and Probe Configurations, resolving issues such as the lack of warning on the frontend when the alert interval was set to 0 minutes.	
Fixed UI issues on the Two-Factor Authentication (2FA) page, addressing problems where the Continue button was enabled without input, and the Cancel button was attempting to submit the form.	
Modified the column type forpem.event_historywhich was restricting the string to 255 characters. The column types foruser_name,component,operation,message, anddetailsin thepem.event_historytable have been modified fromcharacter varying(255)toTEXTto remove string length restrictions, resolving `value too long" errors.	51009
Fixed an issue where pie charts displayed without data did not show an *empty data* message. The API now returns an empty series when all data values are zero.	
Fixed styling issues affecting the buttons in the header of the Audit, Server, and Probe Log Viewer.	
Removed incorrect reference to column *connect_timeout* while calling *pem.server_options* function.	52290
The Barman candle chart now correctly handles and plots in-progress backups (e.g., 'syncing', 'started', and 'waiting for WALS')	
Barman chart is now compatible with dynamic devicePixelRatio. The Barman chart rendering has been updated to be compatible with dynamic changes to thedevicePixelRatio, resolving issues where chart candles were not visible at certain browser resolutions.	
Fixed the *bdr_conflict_history_summary* probe to explicitly cast the *conflict_recorded* value as *bigint* , resolving probe failures caused by decimal places in the value. as bigint.	
A check has been added to the Manage Probes UI to prevent users from disabling system probes that have the *force_enabled* flag set to true.	

continues on next page

Table 11 – continued from previous page

Description	Addresses
Fixed an issue where tooltips in the Barman chart were not showing correct data, particularly when backup data included in-progress statuses.	
Fixed a bug in the Log/Audit Manager where selecting a root node would select all children nodes, even those disabled for selection. all children nodes even if they are disabled.	
Fixed an issue where timezone changes in the job scheduler's 'Schedule' tab were not persisting due to a missing condition in the job schedule macro.	45933
Fixed the size of the preview image displayed under the Theme selection in File > Preferences > Miscellaneous > User Interface, which was previously enlarged.	
Updated log manager component to handle server versions correctly. The log manager component has been updated to correctly support and handle log configurations for EPAS/PG/PGE 18 and above.	
Fixed the issue *string indices must be integers* when attempting to generate an 'Alert History Report'.	53493
Fixed an issue with datetime formatting in Performance Diagnostic charts, resolving the omission of leading zeros on the x-axis (e.g., *4:4:0* instead of *04:04:00*)	
Fixed a UI misalignment issue affecting the Search Box and Delete buttons in the Processes tab button.	
Fixed an issue where email data was not saved for the default email group. Resolved an issue where email notification data was not being saved when the 'Default Email group' was selected for low, high, or medium alert notifications. The backend now correctly defaults to the default email group when no selection is sent.	53339
Fixed an issue where Apache HTTPD's TRACE/TRACK methods were not properly disabled for the HTTPS virtual host Rewrite rules have been moved toedb-ssl-pem.confto ensure these methods return a 403 Forbidden response.	53214, 54529
Fixed a bug where attempting to open dashboards from the context menu (e.g., by right-clicking a server or database) would fail.	
Fixed an issue causing some Pie Charts in the Dashboard to fail with an *Invalid literal for int() with base 10"* error.	
Fixed errors that occurred when editing server connection parameters from a user different than the server owner, ensuring that the server properties update and get APIs handle these scenarios correctly.	
Fixed an issue whereby theunregister-agentcommand did not accept comma-separated lists of hosts or ports Theunregister-agentcommand has been fixed to correctly accept comma-separated lists of hosts or ports, resolving a previous limitation where it only expected a single integer forpem_port.	
Fixed an issue whereby Replication Solution could not be set while registering a server using pemworker. The pemworker registration process for servers now correctly includes the Replication Solution and its corresponding options.	
Fixed an error that occurred when non-superadmin users attempted to access the Query Tool due to incorrect handling of server options for non-superadmin users.	
Fixed an issue where an inappropriate web server address when specify --host with multiple hosts during configuring PEM Fixed the issue where an inappropriate web server address was displayed at the end ofconfigure_pem_server.shoutput when--hostwas specified with multiple hosts.	
Fixed an SQL issue inpem.process_one_alert()function. Explicitly provided column information when inserting a row into thepem.alert_statustable from thepem.process_one_alert()function, resolving an exception (likeERROR: column "state_change_count" is of type integer but expression is of type interval) that occurred after upgrading.	

continues on next page

Table 11 – continued from previous page

Description	Addresses
Fixed an error where agents were not listed correctly when a server was registered into a new server group.	
Fixed an issue where the Connection Manager was not fetching the server connection parameters according to the permissions of the logged-in user.	
Fixed *get_server.sql* which was returning multiple entries if the user detail was not present in the *server_options* table, ensuring a single entry is returned as expected.	

2.5 Postgres Enterprise Manager 10.2.0 release notes

Released: 14 August 2025

New features, enhancements, bug fixes, and other changes in Postgres Enterprise Manager 10.2 include the following:

2.5.1 Highlights

- New alerts for failover events
- Automatically disable redundant alerts on replicas
- PEM can now run in FIPS-enabled environments
- Support for monitoring EDB Postgres Distributed 6

2.5.2 Enhancements

Description	Addresses
Added alert support for EFM/Patroni failover events.PEM now provides a built-in alert for failover in EFM and Patroni clusters.	
The 'Last Vacuum' and 'Last AutoVacuum' alerts are now disabled automatically on replica nodes.PEM will no automatically disable the 'Last Vacuum' and 'Last AutoVacuum' alerts on servers which are currently physical replicas of a primary. This ensures you do not get duplicate alerts in primary/standby clusters.	
Added new probes *bdr_node_group_config_summary* and *bdr_analytic_slot_name* for EDB Postgres Distributed 6; updated the *bdr_node_summary* probe to include the *node_uuid* column.	
Added a new column *parent_table_name* to the *oc_table* probe to indicate partitioned/parent table relationships.	
Introduced new config item *SESSION_DIGEST_METHOD* to support FIPS-enabled environments.The new configuration option *SESSION_DIGEST_METHOD* allows the user to specify the default crytographic hash used for signing data in the PEM web application. The default is 'hashlib.sha1'. In future this will change to 'hashlib.sha256' as sha1 is deprecated. You may change to sha256 today by specifying *SESSION_DIGEST_METHOD = 'hashlib.sha256'* in *config_local.py*. Prior to this change, PEM would reference the MD5 hash library, and even though it wasn't used this would cause PEM to crash on FIPS-enabled systems where OpenSSL doesn't provide MD5.	
Introduced dynamic naming for the Monitoring panel.The text displayed in the tab for monitoring panels is now a descriptive name based on the contents of the panel. This makes it easier to navigate with multiple panels open.	
Enhanced line chart tooltips to display seconds in timestamp values.	
Upgraded HTTPD to 2.4.65 for PEM Windows installer.	

continues on next page

Table 12 – continued from previous page

Description	Addresses
Added self-diagnostic queries to the System Configuration report.	
Enabled Markdown formatting in chart description tooltips.	
PEM will now preserve the SSL port, SSL certificate path, and ServerName when upgrading or reconfiguring PEM.This change allows users to change these parameters to suit their environment without having them overwritten during PEM upgrade.	

2.5.3 Changes

Description	Addresses
Removed MD5 implementation from the PEM agent Removed some unused code from the PEM Agent which implemented the MD5 hash algorithm.	
Renamed heartbeat_interval to heartbeat_tolerance and changed the behavior be easier to understand. In prior versions of PEM, the *heartbeat_interval* setting (a setting for each PEM agent stored in the PEM server) determined how long PEM would wait for a heartbeat until an agent was considered down. Specifically, if there was no heartbeat for *2 * heartbeat_interval + 15 seconds*, the agent was considered down. Fifteen seconds is the gap between heartbeats. The name of the parameter often caused confusion, with users thinking they were setting the heartbeat period. The factor of two in the formula above also caused confusion. We have replaced the setting with *heartbeat_tolerance* and adjusted the formula to *heartbeat_tolerance + 15 seconds* to make the behavior more obvious. When upgrading to PEM 10.2, your previous settings will be respected and converted into an equivalent *heartbeat_tolerance*.	
Removed support for PostgreSQL and EPAS versions older than 11 from the Alternate Code tab in Manage Probes.	
Changed the *unregister-server* CLI option from using the *--server-addr* to *--asb-host-name* for consistency.	

2.5.4 Bug Fixes

Description	Addresses
Resolved server registration issues caused by missing ssl column in the pem.server table (affecting PEM 10.1.1).	
Fixed an issue whereby the 'primary' tag would appear on PGD servers rather than the PGD-specific tags.	
Fixed a validation error when using comma-separated emails in Email Groups.	
Addressed UI issues with MFA in PEM 10. Addressed multiple issues whereby the MFA UI was not rendering correctly and was therefore difficult or impossible to use.	
Fixed duplication of existing entries when adding/deleting new entries in Email Groups.	
Resolved an issue with the Delete button malfunctioning after adding a new Email Group.	
Fixed an issue whereby the Patroni Cluster Status probe could write data to PEM for non-Patroni servers.	

continues on next page

Table 14 – continued from previous page

Description	Addresses
Fixed an issue whereby libpq5 was not installed during PEM configuration despite being required.	
Fixed an issue whereby the Streaming Replication dashboard couldn't be opened on Patroni servers.	
Fixed an issue with Patroni DCS health state reporting where the health status was incorrectly set to false when no leader was found or when the Patroni node list was empty.	
Fixed an issue that disabled toggle buttons and time frequency settings on Manage Alerts	
Fixed an issue where an invalid EFM installation path or cluster name would not result in an error or warning in PEM agent logs.	
Fixed an issue preventing users from modifying data retention settings for system probes via the Manage Probes panel.	
Fixed an issue with the database restriction feature by updating the field name from *db_restriction* to *db_res*.	
Fixed a Windows build issue caused by an unsupported function in the Patroni probe	
Resolved agent registration errors caused by missing agent.cfg, which led to default value issues and unknown agent entries when using force-registration option	
Updated SQL for Pie Chart settings to remove dashboard-level setting support, fixing color update issues	
Fixed an issue which caused an incorrect chart title and missing pie chart color selection during data load.	
Added missing Y-axis labels on Bar Charts.	
Enabled the Download button by default in the Alert History dialog.	
Removed redundant error icons in charts.	
Fixed threshold-related errors in cmtemplate table charts.	
Fixed an issue where errors would appear when chart data was empty.	
Reduced redundant API calls to *server_versions* and added pagination for the Alternate Code table.	
Corrected alert selection UI and toggle switch behavior for disabled webhooks	
Fixed synchronization issues in chart colors, ensuring consistent color application across charts.	
PEM will now stop uWSGI and clean NGINX config when uninstalling PEM in NGINX mode.	
Added libjpeg dependency for PEM server runtime.	
Fixed charts in Streaming Replication dashboard based on Server Replication properties	
Fixed an issue where the audit configuration job was not getting created.	
Fixed an issue where Capacity Manager report generation was failing.	
Fixed an issue with interval calculation in Custom Probes.	
Updated the Patroni timeline mismatch alert template query to handle empty string values properly.	
Fixed an issue where the streaming replication dashboard could be opened even when a server lacks streaming replication data.	
Updated the initial data check API for the streaming replication dashboard to conditionally check different monitoring tables based on the server's replication solution type.	

2.6 Postgres Enterprise Manager 10.1.2 release notes

Released: 21 July 2025

This is a bug-fix release for PEM 10.

!!!Important This release only includes fixes for the PEM agent (all platforms) when used with PEM Server 10.1.1 and later, and the PEM Windows installer.

For this reason, you don't need to apply this patch to existing PEM server installations, or to PEM Agents used with a PEM server version earlier than 10.1.1.

Description	Addresses
Fixed an issue whereby the *pemworker register-server* command would fail with an error.	
Fixed an issue whereby PEM server installation would fail on Windows.	

2.7 Postgres Enterprise Manager 10.1.1 release notes

Released: 7 July 2025

This is a bug-fix release for PEM 10.

2.7.1 Bug Fixes

Description	Addresses
Fixed an issue whereby Performance Diagnostic would not display any data. A bug was preventing PEM from to connecting to the target server to collect data, meaning that the Performance Diagnostics view would always be empty. This has now been resolved and Performance Diagnostic works as intended.	47603,48815,50207
Fixed an issue where shell/batch jobs would not save. Previously, PEM was incorrectly attempting to write a server ID to the backend database when creating a shell/batch job. However, server ID is undefined in this case so the job would not be saved and an error would be displayed.	
Fixed an issue whereby the *patroni_node_status* probe was not created during upgrade from PEM 10.0 to PEM 10.1.	
Fixed an issue whereby the libpq5 package was not installed in cases where it should have been installed.	
Fixed an issue with the v14 API that could result in a 'Circular reference detected' error.	49313
Fixed an issue whereby single quoted strings in job code would give a spurious syntax error.	
Fixed an issue whereby connection parameters were not being saved or loaded in server properties.	
The logic of the *send_email* function is now more robust to unusual *email_group_options*.	
Fixed an issue whereby the PUT method of the /server API endpoint would fail. This issue resulted in an error *Driver.qtLiteral() missing 1 required positional argument: 'conn'*.	
Removed deprecated directive of 'Order deny,allow' from Apache HTTPD configuration.	

continues on next page

Table 16 – continued from previous page

Description	Addresses
Fixed an issue whereby QR codes where not generated on some systems. This issue was resolved by adding a dependency on the requisite *libjpeg* libraries for each system.	

2.8 Postgres Enterprise Manager 10.1.0 release notes

Released: 15 May 2025

New features, enhancements, bug fixes, and other changes in Postgres Enterprise Manager 10.1 include the following:

2.8.1 Highlights

- Support for monitoring and switchover of Patroni clusters
- Automatic tagging of node roles in PGD clusters
- Improved support for HA PEM architectures
- Resolved the issues with Kerberos authentication on RHEL 9

2.8.2 Enhancements

Description	Addresses
Added support for monitoring and switchover of Patroni clusters.PEM 10.1 includes a new 'Patroni Cluster Status' to collect information from Patroni. This information is displayed in the Streaming Replication dashboard for Patroni nodes. Additionally, PEM now includes five server-level alert templates for Patroni nodes:- Patroni cluster paused- Patroni DCS not healthy- Patroni down or out of contact- Patroni no leader detected- Patroni timeline mismatch	
PEM now automatically tags PGD nodes with their role.A new probe 'PGD Node Roles' has been added to PEM. This probe collects the roles of each PGD node. The data collected by the probe is used to automatically label nodes by role.	
Added support for multi-host connection strings from web application to PEM server-When configuring a PEM web application, you may now provide multiple host addresses for the PEM backend. The web application will try each in turn until it finds one that accepts write connections. This should be used in conjunction with a *Query information* .	
Added support for multi-host connection strings PEM Agent to PEM backend database server.When configuring a PEM agent, you may now provide multiple host addresses for the PEM backend. The agent will try each in turn until it finds one that accepts write connections. This should be used in conjunction with a *Query information* .	
Introduced a *pem.register_pem_server(server_id)* function to allow registering a database server as a replica PEM serverWhen configuring a High Availbility PEM architecture, this option may be used to register additional copies of the PEM backend. The function creates static system tasks for all these servers, but they are disabled by default. They will be enabled only for the primary. After promotion, the new primary PEM server will take over the system tasks automatically. This removes the dependency of server/agent id 1 to run the system jobs which existed in prior versions of PEM.	
Added new API version v14	

continues on next page

Table 17 – continued from previous page

Description	Addresses
Introduced a *Steps* sub-panel within the Scheduled Task detail view.This panel displays all steps associated with a task, along with the last run status and execution time of each step, enabling better visibility and monitoring.	
Added *--cluster-name* switch for pemworker to specify cluster name when registering agent/server.	
PEM Agents can now be included in Clusters	
PEM can now call a webhook on job completion/failure.When configuring a webhook endpoint. You may now choose between 'Alert' and 'Job' webhook types. Webhooks of type 'job' will be fired according to the notification settings of each job.	
Added the ability to enable disabled probes from charts that depend on them.	

2.8.3 Changes

Description	Addresses
Merged pgAdmin4 9.2. PEM 10.1 is based on pgAdmin 9.2 and therefore inherits the majority of fixes and features from this release.	
Added ability for PEM to display info messages on charts, for example if data is missing.	
Added pywinpty package to dependencies on Windows.	
Jobs which are created by an alert are no longer listed in the PEM object explorer. Jobs created by alerts were previously visible in object explorer. This led to confusion as users were able to edit these jobs in unintended ways which would typically result in breakage.	

2.8.4 Bug Fixes

Description	Addresses
Fixed an issue whereby an error in a dashboard would prevent all subsequently accessed dashboards from loading correctly.	
Fixed an issue whereby psycopg errors would occur while authenticating PEM using kerberos on RHEL 9PEM is now fully compatible with Kerberos on RHEL 9, so users of this combination are recommended to upgrade to PEM 10.1.	
Fixed an issue whereby dashboards were not updated after being edited.	
Fixed an issue whereby custom probes with special characters in the probe code were not created properly.	47765
Fixed an issue in SQL Profiler whereby 'stop trace' was not working as expected.	
Fixed an issue where probe configurations were not getting saved from UI upon changing the values.	
Fixed an issue whereby EFM probes would fail due to a JSON data parser exception.	46527
Fixed an issue where date in Alert Status table yields year as 1970.	
Fixed an issue with the streaming replication chart where the user would see an error *Column "Cluster Name" does not exist*.	47332
Fixed an issue which could result in an error *The error - module 'lib' has no attribute 'X509_V_FLAG_NOTIFY_POLICY'* during PEM configuration.	
Fixed an issue whereby a "threshold" key error was occurring on update of Alert fields other than threshold values.	

continues on next page

Table 19 – continued from previous page

Description	Addresses
Fixed an issue whereby charts were not automatically refreshed after being modified.	
Fixed an issue where the chart legend was not being accessible.	
Fixed an issue whereby Audit Manager would not display a useful message if no EPAS instances were found.	
Fixed an error that could occur in the browser when performing EFM switchover from PEM.	
Fixed an issue where the Alert Templates page didn't display properly after adding a new template.	
Fixed an issue where Alerts were disappearing from the Manage Alerts panel after selection of objects in cluster type node.	48476
Fixed an issue where special characters in column names caused unexpected behavior in Capacity Manager charts.	
Fixed an issue where the tooltip doesn't go away after moving the pointer off a chart.	
Error catching logic missing in render_as_query function of linechart.	
Improved exception handling when running *agent_ssl_passphrase_script* script.	
Fix for NaN values in PieChart.	
Fixed an issue in Email Groups whereby the delete button would be enabled on the default group when it should be disabled.	
Fixed bug that would cause 'duplicate key' errors in browser console logs.	
Fixed an issue with the webhook 'test' button state whilst waiting for a response.	
Fixed a PEM configuration failure on NGINX/uWSGI when UMASK is 0027	
Fixed an issue whereby NGINX could not be started when ipv6 was disabled	
Fixed an issue whereby Alerts were not saved when execute_alert_on was toggled.	
Fixed an issue with redirects after editing a dashboard.	
Fixed an issue where breadcrumb tree was not displaying properly when a dashboard was opened from the context menu.	
Fixed the missing teams field from the Register/Edit server dialogue. Also added the privilege check while registering the server.	

2.9 Postgres Enterprise Manager 10.0.0 release notes

Released: 24 March 2025

Note

When upgrading to PEM 10 on RHEL-like systems, to prevent the upgrade from being blocked by the removal of the obsolete edb-pem-docs package, use the following dnf command:

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

2.9.1 Highlights

- Support for grouping servers and agents into clusters, with automatic tagging of the server role
- Refreshed, more performant user interface including workspaces to keep your tabs better organized
- Better performance due to optimized deletion of expired data
- Support for NGINX web server

2.9.2 Enhancements

Description	Addresses
Added support for Clusters in Object Explorer.PEM 10 introduces the concept of Clusters. These are groupings of Servers and Agents used to represent a High Availability (HA) cluster, for example a cluster using Postgres Distributed or EDB Failover Manager. You can create and populate Clusters in the Object Explorer. In future releases we will expand this functionality to improve the experience of working with HA clusters in PEM.	
Added support for tagging servers and agents.PEM now supports arbitrary tagging of servers. Presently this is purely a cosmetic feature. In the future we will expand it to support filtering and selection by tags.	
Ported the UI to the React framework with numerous useability improvements.The entire PEM user interface has been ported to the React framework. This brings the PEM project back into line with pgAdmin, making it easier for us to ull enhancements from the upstream. It also delivers a more modern and performant interface. We have taken advantage of this porting to make numerous improvements to the UI including:- Added pagination in Probes/Alerts.- Implemented workspaces.- Divided scheduled tasks into Task and History tabs.- Implemented a dialog view for creating Alerts, Probes and other objects, replacing the grid view.	
Incorporated recent enhancements from pgAdmin.The updated architecture of PEM 10 has allowed us to pull many enhancements from pgAdmin including:- Generate Entity Relationship Diagrams- Plot charts in Query Tool results- Show real-time system metrics dashboard using the systemstats extension	
Added *read_operations* and *write_operations* columns for IO Analysis probe.This data can be used to monitor and alert on IOPS (Input/Output Operations Per Second), which is of particular relevance when using cloud storage.	
Added a *node_type* column to the *server_info* probe.The *node_type* may take one of the following values: *primary*, *replica*, *readonly*, *standalone*.	
Separated PEM tabs into workspaces for easier management.Panels/Tabs for the following PEM components will be opened in the 'Enterprise Manager' workspace:- Alert templates- Email group- Webhooks- Custom probes- Manage ChartsPanels/Tabs for the following PEM components will be opened in the 'Enterprise Tools' workspace:- SQL Profiler- Performance diagnosis- Capacity manager reportPanels/Tabs for the following PEM components will be opened in the 'Object Explorer' workspace:- Probe configuration- Alerts	
Added support for NGINX as the default web server.PEM 10 supports both Apache HTTPD and NGINX as the web server on Linux. On Windows only Apache HTTPD is supported. The default for new installations is NGINX. Upgrades from PEM 9 will continue to use Apache HTTPD unless you opt to switch the web server by running the */usr/edb/pem/bin/switch-web-server.sh* script.	
Added support for 'minimum query duration' in SQL Profiler.You can now specify a minimum query duration when creating a trace in SQL Profiler. This allows you to only trace long-running queries, which in turn means you can run a trace for a much longer period without exceeding the size limit.	
Added Ubuntu 24.04 support.	
The PEM agent will now wait for a while after connecting to the PEM server to avoid false alerts.This is configurable via the *alert_reconnect_delay* parameter in *agent.cfg*.	
Added support for EFM 5.0; 'xlog *' columns are renamed as 'lsn* '.	
PEM will automatically add tags showing whether a server is a primary or a replica.	
Improved the detailed info SQL for the global alert templates 'Servers Down' and 'Agents Down'.	

continues on next page

Table 20 – continued from previous page

Description	Addresses
Improved the purge history function performance.	

2.9.3 Changes

Description	Addresses
Log Analysis Expert, Tuning Wizard, Postgres Expert, and Index Advisor are removed.These features have been removed from PEM 10.0 onwards. For Postgres tuning we recommend the EDB Postgres Tuner extension and the accompany web utility . The Index Advisor extension for EDB Server has been deprecated and replaced by EDB Query Advisor , which is compatible with all Postgres distributions. For log analysis we recommend pgBadger .	
The PEM backend database is no longer automatically tuned.We now recommend using EDB Postgres Tuner to tune the PEM backend database.	
Fixed an issue where mount points are missing from the io_analysis probe.	
Fixed a bug affecting the Operating System dashboard whereby the network traffic information was not shown.	40938

2.9.4 Bug Fixes

Description	Addresses
Fixed an issue where a user with *pem_manage_dashboard* is not able to see the custom dashboard.	
Fixed an issue where mount points are missing from the io_analysis probe.	
Fixed a bug affecting the Operating System dashboard whereby the network traffic information was not shown.	40938

CHAPTER 3

Platform compatibility

For information about the platforms and versions supported by PEM, see Platform Compatibility .

Note

Postgres Enterprise Manager 8.3 and later is supported on SLES.

CHAPTER 4

Hardware requirements for installing the PEM server on Linux

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

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

Additional disk space is required for data storage. Resource use varies based on the probes that are defined and enabled and the activity level on the monitored databases. Monitoring server resources as you use PEM lets you know when you need to expand your initial system configuration.

CHAPTER 5

Understanding Postgres Enterprise Manager components and architecture

Postgres Enterprise Manager (PEM) monitors and manages multiple Postgres servers through a single graphical interface. PEM can monitor the following areas of the infrastructure:

- **Hosts** _ One or more servers (physical or virtual) and their operating systems.
- **Database servers** _ One or more instances of PostgreSQL, EDB Postgres Advanced Server, or EDB Postgres Extended Server (formerly known as 2ndQPostgres) running on a host.
- **Databases** _ One or more databases and their schema objects, such as tables and indexes.

> **Note**
>
> The term Postgres refers to PostgreSQL, EDB Postgres Advanced Server, or EDB Postgres Extended Server.

PEM consists of individual software components:

- **PEM server** _ The PEM server is the data repository for monitoring data and a server to which agents and clients connect. The PEM server consists of an instance of PostgreSQL, an associated database for storing monitoring data, and a server that provides web services.
- **PEM agent** _ The PEM agent is responsible for executing tasks and reporting statistics from the agent host and the monitored Postgres instances to the PEM server. A single PEM agent can monitor multiple installed instances of Postgres that reside on the same host where the agent is installed or on multiple remote hosts. You can install an agent as part of the PEM server installation or you can install a stand-alone agent.
- **PEM web client** _ The PEM web interface allows you to manage and monitor Postgres servers and use PEM extended functionality. The web interface software is installed with the PEM server and is accessed using any supported web browser.
- **SQL Profiler** _ SQL Profiler is a Postgres server extension to record the monitoring data and query plans for the SQL Profiler tool to analyze in PEM. This is an optional component of PEM, but the extension must be enabled in each instance of Postgres for which you want to use it. You can use the SQL Profiler with any supported version of an EDB distribution of Postgres, not just those managed through the PEM server. See *Installing the SQL Profiler extension* for details and supported versions.

5.1 PEM architecture

The following architectural diagram shows the relationships between the PEM server, clients, and managed and unmanaged Postgres servers.

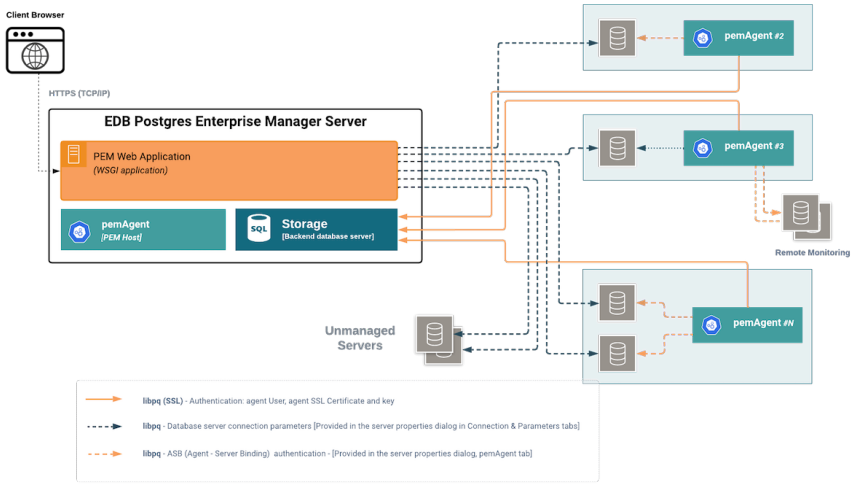


Fig. 1: PEM Architecture

5.1.1 PEM server

The PEM server consists of an instance of Postgres, a web server providing web services to the client, and a PEM Agent. PEM uses a server-side cryptographic plugin to generate authentication certificates.

The instance of Postgres and the web server can be on the same host or on separate hosts.

> **Note**
>
> All the PEM features are available on either backend database server you select: PostgreSQL or EDB Postgres Advanced Server.

- **Postgres instance (database server)** _ This is the backend database server. It hosts a database named `pem` , which acts as the repository for PEM server. The `pem` database contains several schemas that store metric data collected from each monitored host, server, and database.
- `pem` _ This schema is the core of the PEM application. It contains the definitions of configuration functions, tables, or views required by the application.
- `pemdata` _ This schema stores the current snapshot of the monitored data.
- `pemhistory` _ This schema stores the historical monitored data.
- **Web server (Nginx/HTTPD)** _ The PEM web application is deployed as a WSGI application behind a web server to provide web services to the client. It is made up of the following:
- **Web content presentation** _ The presentation layer is created by the web application (such as browser and login page).
- **Rest API** _ The REST API allows integration with other apps and services.
- **Database server administration/management** _ You can perform database server administration and management activities like CREATE, ALTER, and DROP for managed and unmanaged servers.

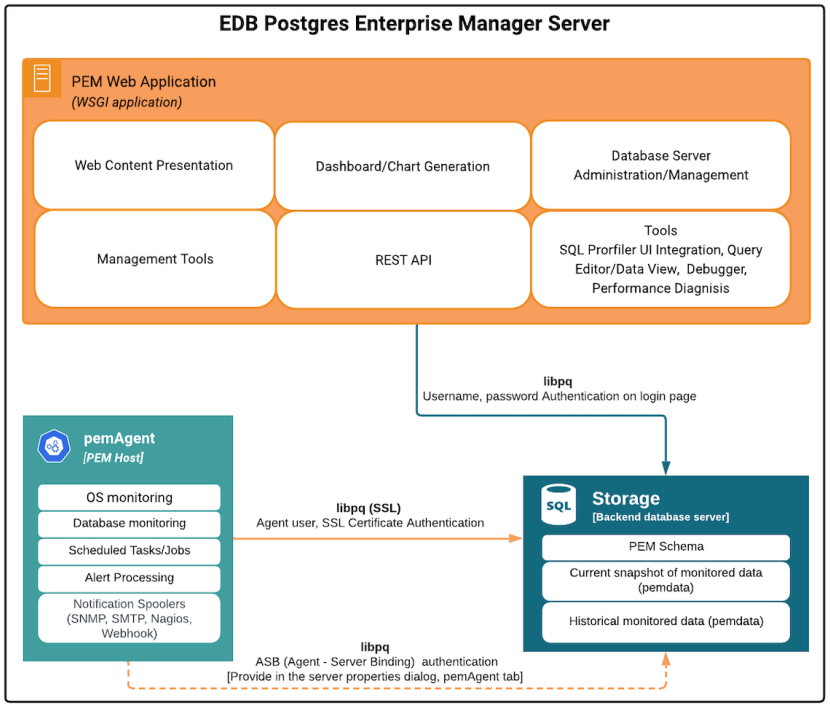


Fig. 2: PEM Server

- **Dashboard/chart generation** _ Internally, the web application includes functionality that generates dashboards and charts.
- **Management tools** _ The Audit Manager, Capacity Manager, and Log Manager are available in the web application.
- Other tools provide functionality on managed or unmanaged servers: - **SQL Profiler UI integration** _ SQL Profiler generates easily analyzed traces of session content. - **Query editor/data view** _ The Query editor allows you to query, edit, and view data. - **Debugger** _ The debugger helps you debug queries. - **Performance diagnostics** _ Performance diagnostics help you analyze the performance of Postgres instances.

We recommend that you use a dedicated machine to host production instances of the PEM backend database. The host might be subject to high levels of data throughput, depending on the number of database servers that are being monitored and the workloads the servers are processing.

5.1.2 PEM agent

The PEM agent is responsible for collecting monitoring data from the machine and operating system and from each of the Postgres instances to which they are bound. Each PEM agent can monitor one physical or virtual machine and is capable of monitoring multiple database servers locally that are installed on other systems. These servers can be installed on the same system or remotely. It's also responsible for executing other tasks that the user might schedule such as server shutdowns, SQL Profiler traces, and custom jobs.

A PEM agent is installed by default on the PEM server when you install the PEM server. It is generally referred to as a PEM agent on the PEM host. Separately, you can also install the PEM agent on the other servers hosting the Postgres instances you want to monitor using PEM.

Whether monitoring locally or remotely, the PEM agent connects to the PEM server using PostgreSQL's libpq, using SSL certificate-based authentication. The PEM agent installer in Windows and pemworker CLI in Linux is responsible

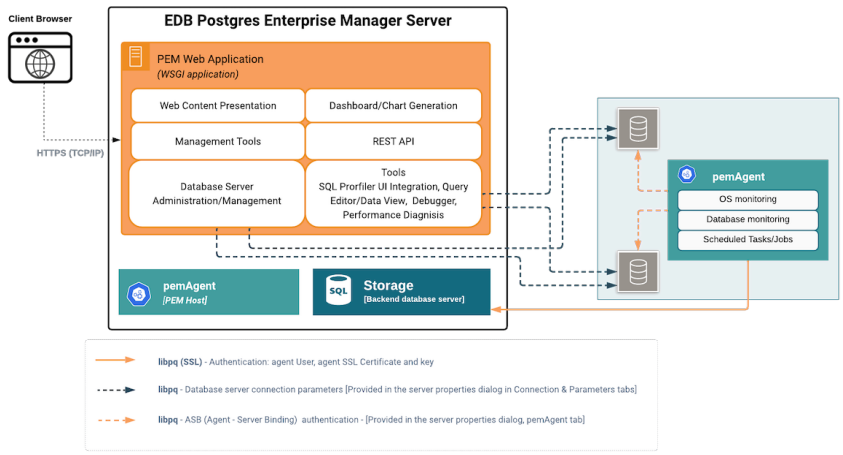


Fig. 3: PEM Agent

for registering each agent with the PEM server and generating and installing the required certificates.

There is only one-way traffic between the PEM agent and PEM server. The PEM agent always connects to the PEM server.

The PEM agent must be able to connect to each database server that it monitors. This connection is made over a TCP/IP connection (or, optionally, a Unix Domain Socket on Unix hosts), and can optionally use SSL. You must configure the connection and authentication to the monitored server.

Once configured, each agent collects statistics and other information on the host and each database server and database that it monitors. Each piece of information is known as a *metric* and is collected by a *probe*. Most probes collect multiple metrics at once for efficiency. Examples of the metrics collected include:

- Disk I/O statistics
- Network statistics
- Database server version string
- Database server configuration option (GUC) values
- Table access statistics
- Table and index sizes

For a list of PEM probes, see *Probes* .

By default, the PEM agent bound to the database server collects the OS/database monitoring statistics and also runs any scheduled tasks/jobs for that particular database server, storing data in the `pem` database on the PEM server.

The alert processing, SNMP/SMTP spoolers, and Nagios spooler data is stored in the `pem` database on the PEM server and is then processed by the PEM agent on the PEM host by default. However, you can enable processing by other PEM Agents by adjusting the SNMP/SMTP and Nagios parameters of the PEM agents.

For more information about these parameters, see *Server configuration parameters - reference* .

5.1.3 PEM web application

The web application connects to the PEM server and allows direct management of managed or unmanaged servers and the databases and schemas that reside on them.

The web application allows you to use PEM functionality that makes use of the data logged on the server through features such as dashboards and Capacity Manager.

5.1.4 SQL Profiler extension

You don't have to install the SQL Profiler extension on every server, but you must install and enable the extension on each server on which you want to use SQL Profiler. You might also want to install and enable the SQL Profiler extension on unmonitored development servers. You can also temporarily install the SQL Profiler extension for ad hoc use.

You can use SQL Profiler on servers that aren't managed through PEM. However, to perform scheduled traces, a server must have the extension installed and enabled and must be managed by an installed and configured PEM agent.

For more information, see *Installing the SQL Profiler extension* and *Using SQL Profiler* .

CHAPTER 6

Deployment considerations

Before deploying Postgres Enterprise Manager, consider these factors.

Considerations	Implementation instructions
What specification of servers do I need to run PEM?	*Sizing PEM deployments*
Does my EDB subscription cover my planned use of PEM?	*Query information*
Is a standalone server sufficient or do I need a high availability architecture?	Installing the server or *Query information*
Do I need to implement connection pooling?	*Query information*
How will I manage generation and renewal of SSL certificates?	*Deciding on an SSL model*
What type of user authentication should I use?	*Authentication options*
What actions should I take to avoid security vulnerabilities?	*Securing your deployment*
Where should I host the web server?	*Query information*

6.1 How PEM licensing works

PEM is intended to be used for monitoring and managing Postgres instances supported by EDB. As such, if you have a subscription that entitles you to use PEM, adding it to your stack to monitor those instances generally incurs no additional license costs.

6.1.1 Licensing rules

To use PEM, you must have an EDB Standard or Enterprise subscription.

Any Postgres servers you monitor using PEM must be covered by an EDB Standard or Enterprise subscription. This is true whether you're installing the PEM agent locally, through remote monitoring, or by making a client connection from the PEM web application.

Non-Postgres servers that don't otherwise consume cores (for example PGD Proxy nodes or machines running Barman) may be monitored by PEM. No additional cores are consumed by monitoring those servers with PEM.

The Postgres instance used as the PEM backend doesn't consume cores from your subscription. Likewise, no cores are consumed by any other PEM components. All these components are covered by your support agreement.

6.1.2 Examples

Adding PEM to a fully supported environment

Suppose a customer with an Enterprise subscription for 32 cores has 8 x 4-core servers running EDB Postgres Advanced Server, thereby fully consuming their 32 cores. This customer can install PEM on a fifth server and use it to monitor their 8 EDB Postgres Advanced Server servers. This configuration requires no change to their subscription, as the PEM server doesn't consume cores, and the monitored EDB Postgres Advanced Server instances are already fully covered.

Likewise, if this customer added a Barman server and connected it to PEM for monitoring, they would consume no additional cores.

Adding unsupported servers to PEM

Suppose a customer with a Standard subscription for 36 cores has PEM and 6 x 6-core servers running PostgreSQL covered by EDB support, consuming their 36 cores, as in the previous example.

The customer also has 10 x 2-core PostgreSQL servers that aren't covered by any EDB subscription. These servers won't be monitored by PEM, as they aren't covered by a Standard or Enterprise subscription. If the customer wants to monitor these servers, they must add 20 more cores to their subscription.

Note

An EDB Community 360 subscription for the additional 20 cores isn't sufficient. Servers monitored by PEM must be covered by Standard or Enterprise.

6.2 PEM server and agent connection management mechanism

You can manage the connections effectively either through a moderate number or with connection pooling in Postgres Enterprise Manager (PEM).

You can use PgBouncer as a connection pooler for limiting the number of connections from the PEM agent to the PEM server on non-Windows machines.

You can see below how PEM agents manage database connections both with and without the use of PgBouncer as a connection pooler.

6.2.1 Without PgBouncer

In the default configuration, each PEM agent connects to the PEM database server directly using SSL for encryption. Each PEM agent uses its own dedicated user for the connection.

For example, a user agent with ID 1 connects to the PEM database server as agent1.

6.2.2 With PgBouncer

In a PgBouncer-enabled environment, PEM agents can't connect to the PEM database server directly. PEM agents must use a proxy user that you configure specifically for the connection to PgBouncer. In the example, the proxy user handling all PEM agent connections is called `pem_agent_user1` .

Once the PEM agents connect to PgBouncer using SSL, PgBouncer is responsible for managing connection requests to the PEM database server. PgBouncer uses the rules you configured for connection pooling to manage the incoming connection requests, for example, by respecting the established maximum number of active connections.

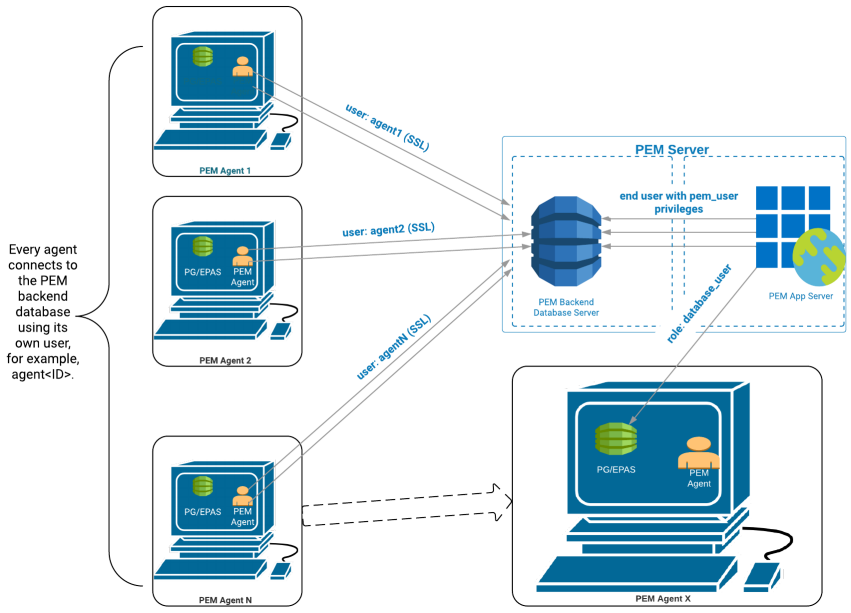


Fig. 1: Connecting to the PEM database without PgBouncer

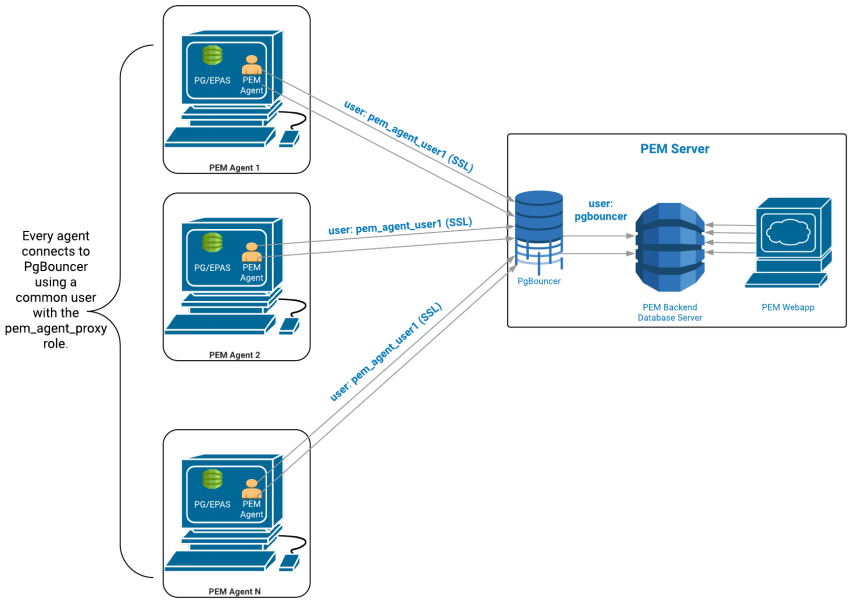


Fig. 2: Connecting to pgBouncer.

!!!note Use PgBouncer version 1.9.0 or later as the connection pooler. Versions 1.9.0 or later support cert authentication.

6.3 Authentication options

PEM user authentication is delegated to the Postgres backend server. To be able to login to PEM with a given username, there must exist a Postgres user of that name in the backend. Users must have the `pem_user` role and be permitted by `pg_hba.conf` to connect to the `pem` database from the web application host. For more on PEM user roles see *Managing PEM authentication* .

By default, PEM requires users to authenticate using a password. You can optionally enforce 2FA, or enable password-less authentication using Kerberos.

6.3.1 1. Two-factor authentication (2FA)

PEM supports 2FA to provide an additional layer of security beyond standard login credentials. It can be configured by modifying parameters in the `config_local.py` file.

- **Supported methods:**
  - **Email** _ Sends a one-time verification code to the user's email address.
  - **Authenticator App** _ Uses TOTP-based applications like Google Authenticator.
- **Key configurations:**
  - **MFA_ENABLED** _ Set to `true` to enable the feature.
  - **MFA_FORCE_REGISTRATION** _ When `true` , you must register for 2FA upon your next login.
- **Mail server:**

For email-based 2FA, PEM can use its internal SMTP configuration or Flask-Mail.

- **Documentation:**

Configuring two-factor authentication .

6.3.2 2. Kerberos authentication

Kerberos provides a secure, ticket-based authentication mechanism for SSO environments. This configuration relies heavily on hostnames and DNS resolution across the Kerberos server, PEM server, and client machines.

Linux-based kerberos

This method involves installing a Kerberos authentication server and configuring principals for both the PEM web application (HTTP) and the backend database (Postgres).

- **Keytabs** _ Security keytabs are extracted from the Kerberos server and deployed to the PEM server to facilitate encrypted communication.
- **Authentication** _ Users obtain tickets via the `kinit` utility. The pg_hba.conf file is updated to use the gss (Generic Security Service) method.
- **Documentation** _ *Configuring Kerberos authentication on Linux* .

Windows active directory (SSPI)

PEM can integrate with Windows Active Directory domain services for Kerberos authentication.

- **Service mapping** _ Active Directory users are created and mapped specifically to HTTP and Postgres service principals.

- **Keytab generation** _ The Windows `ktpass` utility is used to generate keytabs for Linux-based PEM servers to interact with the Windows domain.
- **Browser requirements** _ Client browsers (Firefox/Chrome) must be configured to trust the PEM server URI for SPNEGO/Kerberos negotiation.
- **Documentation** _ *Configuring kerberos authentication on Windows* .

6.4 Deciding on an SSL model

PEM agents are Postgres clients that connect directly to the PEM backend database using libpq, as detailed in the *PEM architecture* .

To securely manage and encrypt communications, PEM leverages the native SSL/TLS capabilities of its Postgres backend.

For this security model to function, the following components must be configured:

- **Agent Authentication**: Every PEM agent must have a client certificate that matches the name of its assigned Postgres user.
- **Root Trust**: The backend Postgres server must be configured with a Root CA file to validate the incoming agent certificates.
- **Server Encryption**: The backend Postgres server requires its own SSL certificate and private key to encrypt the data stream and establish its identity.

There are different models you can use to provide these certificates and keys, depending on your needs and policies. The information below will help you decide what model you wish to follow before starting installation.

6.4.1 Certificate management models

You can configure PEM to handle security certificates in two ways: **Automated (via sslutils)** or **Manual (using an external CA)**. Choosing the right path depends on your infrastructure complexity and compliance needs.

- **Automated with sslutils**: By leveraging the sslutils extension, PEM acts as its own Certificate Authority (CA). When an agent registers, PEM automatically generates the required client certificates.

Best for: Standard, non-HA deployments where speed and convenience are the priority.

Drawback: Offers less control for organizations with strict external CA policies.

- **pgBouncer: If using pgBouncer, sslutils is generally not recommended because pooling requires a single shared user (see below). .. ::**
  - **External CA (manual)**: Instead of relying on sslutils, you can provide certificates signed by your organization's approved CA. When bringing your own certificates, you have two architectural choices:

1. **Common Agent User (shared certificate)**: All agents share a single Postgres user and the same client certificate/key pair.

Pros: The only compatible method for pgBouncer (since transactions must come from the same user to be pooled); much easier to manage fewer certificates.

Cons: Harder to audit specific agent connections; revoking the certificate disconnects all agents.

2. **Unique agent users (individual certificates)**: Every agent has its own unique certificate and a corresponding numbered Postgres user (e.g., agent1, agent2).

Pros: High auditability; revoking one certificate only affects a single agent.

Cons: Significant management overhead due to the large volume of certificates.

6.5 Securing your deployment

To harden your PEM deployment against attack, consider the following measures:

- Ensure PEM, your operating system, and third-party libraries are regularly updated. Without the most recent security patches, your system is vulnerable to cyberattacks. See *Dependencies of the PEM Server and Agent on Linux* to learn more about the system packages used by PEM.
- Ensure the Postgres instance used as the PEM server is kept up to date and apply Postgres security best practices .
-

Secure the web server .

- Configure the *security settings of the PEM web application* as appropriate.

6.5.1 Web server security configuration

You can configure the security of the PEM web server.

On Windows, the supported web server is Apache HTTPD. Apache HTTPD is bundled with PEM under the name PEM HTTPD. The Apache HTTPD configuration file is `pem.conf` , and the SSL configuration file is `httpd-ssl-pem.conf` . Both configuration files are in the `<Apache_Installation_Path>/conf/addons` directory.

On Linux, both NGINX and Apache HTTPD are supported. The NGINX configuration file is `/etc/nginx/conf.d/edb-pem.conf` on RHEL-like systems and `/etc/nginx/sites-available/edb-pem.conf` on Debian-like systems. The Apache HTTPD configuration file is `edb-pem.conf` , and the SSL configuration file is `edb-ssl-pem.conf` . Both configurations files are in the `<Apache_Installation_Path>/conf.d` directory.

Recommendations applied by default

These recommendations are applied by default in new installations of PEM. If you customized your web server configuration or carried it over from a much older version of PEM, you can use this information to verify that your configuration meets current standards.

Disable insecure SSL and TLS protocols

In new installations of PEM, SSL versions SSLv2, SSLv3, TLS 1, and TLS 1.1 are disabled by default. These versions are the most vulnerable and have cryptographic concerns.

For NGINX, PEM adds the following line to the configuration file:

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

For Apache HTTPD, PEM adds the following lines to the SSL configuration file:

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

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

You can verify that TLS 1.1 is disabled using the following command. Replace the URL with your web server's. A return value of 35 means TLS 1.1 is disabled. 0 means it's enabled.

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

Disable web server information exposure

In new installations of PEM, the web server is configured to minimize the information about the server exposed to clients by disabling server tokens and server signatures. Server tokens expose information about the server in response headers. Server signatures expose information in the footers of server-generated pages such as error messages.

For NGINX, PEM adds the following line to the configuration file:

```
server_tokens off;
```

For Apache HTTPD, PEM adds the following lines to the SSL configuration file:

```
ServerTokens Prod
ServerSignature Off
```

Disable directory listing

The directory listing allows an attacker to view the complete contents of directories from which content is served. This listing might lead to the attacker reverse engineering an application to obtain the source code, analyze it for possible security flaws, and discover more information about an application.

To avoid this risk, PEM disables directory listing.

For NGINX, PEM sets `autoindex: off`.

For Apache HTTPD, PEM sets the `Options -Indexes` directive:

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

Cross-site tracing

The TRACE and TRACK HTTP methods are used for debugging servers. When an HTTP TRACE request is sent to a supported web server, the server responds and echoes the data passed to it, including any HTTP headers. We recommend that you disable these methods in the Apache configuration.

In NGINX, TRACK and TRACE methods are disabled by default. In Apache HTTPD, PEM includes the following lines in the configuration file to reject these methods. Some scanners don't understand this syntax and may incorrectly report that these methods are allowed.

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

You can verify that TRACK and TRACE are disabled with the following commands. Replacing the URL with your web server's. A return value of 35 means TLS 1.1 is disabled. 0 means it's enabled. If the methods are disabled, the command returns an HTML response that includes the text `405 Method Not Allowed` or similar.

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

Optimize HTTP headers for security

PEM sets various HTTP header options to improve security. These settings are defined in the `config.py` and `config_distro.py` files. This file is located at `/usr/edb/pem/web` on Linux and at `C:\ProgramFiles\edb\pem\server\share\web` on Windows.

If you want to alter any of these settings, don't edit these files. Instead create (or edit if it already exists) a file named `config_local.py` in the same location and add your desired settings. These settings override those in the `config.py` and `config_distro.py` files. They aren't overwritten during a PEM upgrade.

For detailed information on the `config.py` file, see *Managing configuration settings* .

X-Frame-Options

X-Frame-Options indicates whether a browser is allowed to render a page in an <iframe> tag. It specifically protects against clickjacking. PEM has a host validation `X_FRAME_OPTIONS` option to prevent these attacks, which you can configure in the `config_local.py` file. The default is:

```
X_FRAME_OPTIONS = "SAMEORIGIN"
```

Content-Security-Policy

Content-Security-Policy is part of the HTML5 standard. It provides a broader range of protection than the X-Frame-Options header, which it replaces. It's designed so that website authors can whitelist domains. The authors can load resources (like scripts, stylesheets, and fonts) from the whitelisted domains and also from domains that can embed a page.

PEM has a host validation `CONTENT_SECURITY_POLICY` option to prevent attacks, which you can configure in the `config_local.py` file. The default is:

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

Strict-Transport-Security

The Strict-Transport-Security (HSTS) response header can prevent a man-in-the-middle attack. When you enable the option, websites or web applications tell browsers that they accept only HTTPS and not HTTP. The default is:

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

Note

Adding this parameter can cause problems if config is changed. Therefore, we recommend that you add it only after PEM installation is complete and tested.

X-Content-Type-Options

The X-Content-Type-Options response HTTP header is a marker. The server uses the marker to indicate that the MIME types advertised in Content-Type headers can't be changed and followed. The following is a way to opt out of MIME type sniffing, that is, to say that the MIME types are deliberately configured. The default is:

```
X_CONTENT_TYPE_OPTIONS = "nosniff"
```

X-XSS-Protection

Cross-site scripting (XSS) is one of the most common application layer vulnerabilities in the web servers. XSS enables attackers to inject client-side scripts into web pages that other users view. The HTTP X-XSS-Protection response to the header is a feature of Internet Explorer, Chrome, and Safari. It stops pages from loading when they detect reflected cross-site scripting (XSS) attacks. These protections are unnecessary in modern browsers when sites implement a

strong Content-Security-Policy that disables the use of inline JavaScript ('unsafe-inline'). However, these protections can still provide protections for users of older web browsers that don't yet support CSP. The default is:

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

Cookie security

Cookies are small packets of data that a server sends to your browser to store configuration data. The browser sends them and all other requests to the same server, so it's important to know how to secure cookies. Multiple configuration options in `config.py` can make cookies secure. These are the three most important options:

- SESSION_COOKIE_SECURE _ The flag prevents cookies from sending over an unencrypted connection. The browser can't add the cookie to any request to a server without an encrypted channel. The browser can add cookies only to connections such as HTTPS. The default is:

```
SESSION_COOKIE_SECURE = True
```

- SESSION_COOKIE_HTTPONLY _ By default, JavaScript can read the content of cookies. The `HTTPOnly` flag prevents scripts from reading the cookie. Instead, the browser uses the cookie only with HTTP or HTTPS requests. Hackers can't exploit XSS vulnerabilities to learn the contents of the cookie. For example, the `sessionId` cookie never requires being read with a client-side script. So, you can set the `HTTPOnly` flag for `sessionId` cookies. The default is:

```
SESSION_COOKIE_HTTPONLY = True
```

- ENHANCED_COOKIE_PROTECTION _ When you set this option to `True` , then a token is generated according to the IP address and user agent. In all subsequent requests, the token recalculates and compares to the one computed for the first request. If the session cookie is stolen and the attacker uses it from another location, the generated token is different. In that case, the extension clears the session and blocks the request. The default is:

```
ENHANCED_COOKIE_PROTECTION = True
```

> **Note**
>
> This option can cause problems when the server deploys in dynamic IP address hosting environments, such as Kubernetes or behind load balancers. In these cases, set this option to *False* .

To apply the changes, restart the web server.

For detailed information on the `config.py` file, see *Managing configuration settings* .

Additional recommendations that can be applied manually

These recommendations aren't applied automatically because they require additional information or action specific to the environment in which PEM is deployed.

Secure HTTPD with SSL certificates

During PEM configuration, a self-signed certificate is generated to secure traffic between the web server and clients. To enhance security and to prevent browser warnings that the site isn't secure, we recommend that you replace this certificate with one signed by a trusted certificate authority .

Run the web server from a non-privileged user account

On Linux, PEM uses web server packages provided by the OS. Typically, these create a service unit that runs the web server as the root user.

Running the web server as a root user can create a security issue. We recommend that you run the web server as a unique non-privileged user. Doing so helps to secure any other services running during a security breach.

On Windows, the `WSGIDaemonProcess` directive and features aren't available, so both the web server and the WSGI app run as the system user (the `LocalSystem` account).

It's a good practice to restrict access to the web server to the smallest set of IP addresses compatible with your business needs. This is most commonly done at the network infrastructure level, for example, through firewall configuration, but can also be enforced by the web server. The PEM application configuration file (`<PEM_INSTALLATION_PATH>/web/config_local.py`) supports an `ALLOWED_HOSTS` configuration parameter for this purpose.

For example:

```
#  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]
```

To apply the application configuration file changes, restart the web server.

6.5.2 PEM application security configurations

Session timeout

Setting session expiration time too long in the web application increases the exposure of other session-based attacks. The attacker has more time to reuse a valid session ID and hijack the associated session. The shorter the session interval is, the less time an attacker has to use the valid session ID. To avoid this security issue, we recommend that you set the inactivity timeout for the web application to a low value.

In PEM, you can set the timeout value for a user session. When there's no user activity for a specified duration on the web console, PEM logs the user out of the web console. A PEM administrator can set the length of time for inactivity. This value is for the whole application, not for each user.

To configure the timeout duration, modify the `USER_INACTIVITY_TIMEOUT` parameter in the `config_local.py` file in the `<PEM_INSTALLATION_PATH>/web` directory. By default, this parameter is disabled. Specify the value in seconds.

For example, to specify for an application to log a user out after 15 minutes of inactivity:

```
USER_INACTIVITY_TIMEOUT = 900
```

To apply the change, restart the Apache service.

For detailed information on the `config.py` file, see *Managing configuration settings* .

RestAPI header customization

You can customize the RestAPI token headers to meet your requirements. The default values aren't exposed by the `config.py` file. In the `config_local.py` file, customize the following headers.

PEM_HEADER_SUBJECT_TOKEN_KEY

This configuration option lets you change the HTTP header name to get the generated token. By default, when you send a request to create a token, the server response has an `X-Subject-Token` header. This header contains the value of a newly generated token. If you want to customize the header name, then you can update the `config_local.py` file:

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

This command produces the following output:

```
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

This configuration option lets you change the header name of the HTTP request. With this header name, you can send the token to the PEM server. By default, when you send a request to generate a token, the token header name is `X-Auth-Token` . If you want to customize the RestAPI request header name, you can update the `config_local.py` file:

```
PEM_HEADER_TOKEN_KEY = Pem-Token
```

This setting lets you send the token:

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

PEM_TOKEN_EXPIRY

This configuration option lets you change the PEM RestAPI token expiry time after it's generated. By default, the token expiry time is set to 20 minutes (1200 seconds). For example, to change the token expiry time to 10 minutes, update the `config_local.py` file as follows:

```
PEM_TOKEN_EXPIRY = 600
```

To apply the change, restart the Apache service.

Role-based access control in PEM

Role-based access control (RBAC) restricts application access based on a user's role in an organization. It's one of the primary methods for access control. The roles in RBAC refer to the levels of access to the application for users. Users are allowed to access only the information needed to do their jobs. Roles in PEM are inheritable and additive rather than subscriptive. In other words, as a PEM admin, you need to grant the lowest level role to the user and then grant the roles the user needs to perform their job. For example, to give access only to SQL Profiler:

```
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;
```

For detailed information on roles, see *PEM roles* .

Password management

One security tip for PEM administrative users is to regularly change your PEM login passwords to something new. Changing your password:

- Prevents breaches of multiple accounts
- Prevents constant access
- Prevents the use of saved passwords on a physically unsecured system
- Limits access gained by keystroke loggers

Run PEM agent jobs with a non-root user

In most cases, the PEM agent is installed as a root user and runs as a daemon process with root privileges. By default, PEM disables running the scheduled jobs/task. PEM provides support for running scheduled jobs as a non-root user by changing the PEM agent configuration file.

To run scheduled jobs as a non-root user, modify the entry for the `batch_script_user` parameter in the `agent.cfg` file and specify the user to run the script. You can specify either a non-root user or root user identity. If you don't specify a user or the specified user doesn't exist, the script doesn't execute.

After modifying the file, restart the agent. If a non-root user is running the PEM agent, the value of `batch_script_user` is ignored. The same non-root user used for running the PEM agent executes the script.

To invoke a script on a Windows system, set the registry entry for `AllowBatchJobSteps` to `true` and restart the PEM agent. PEM registry entries are located in:

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

Changing the PEM agent and PEM backend database server certificates

By default, when you install PEM, the installer generates and uses self-signed certificates for the PEM agent and PEM database server. The PEM agent uses these certificates when connecting to the PEM database server. To use your own SSL certificate for the PEM agent and PEM database server, see Managing certificates .

Note

PEM doesn't support placing the SSL CA certificates at a custom location. Don't change the location of *ca_certificate.crt* and *ca_key.key* .

6.6 Web server installation options

While installing the PEM server, you can specify your hosting preferences for the web server. For production environments, best practice is to have the PEM server and web server on separate hosts.

6.6.1 PEM server and web server on separate hosts

1. Install the PEM server on both the hosts. See *Installing the PEM server and PEM-HTTPD on separate hosts* .
2. Configure the PEM server host by selecting the **Database** option on the first host.
3. Configure a web server by selecting the **Web Services** option on the second host.

For more information about configuring a PEM server, see *Configuring the PEM server on Linux* .

6.6.2 PEM server and web server on the same host

1. Install the PEM server. See *Installing the PEM server and PEM-HTTPD on separate hosts* .
2. Run the configuration script. To install the PEM server and web server on the same host, select the **Web Services and Database** option. See *Configuring the PEM server on Linux* .

6.7 Sizing PEM deployments

It's important to allocate the right amount of resources to your PEM servers and understand the key factors driving resource requirements in PEM. T-shirt sizing helps you get started.

!!!Important PEM is Postgres PEM is fundamentally built upon a Postgres database, with most of its core logic executed via SQL functions. It's crucial to treat PEM as a standard Postgres instance. To ensure optimal responsiveness, adhere to all conventional Postgres configuration and tuning best practices. PEM also monitors performance. You can use its integrated tools, such as dashboards, the *Performance Diagnostic* , and *Upgrading SQL Profiler* , to analyze and understand its behavior.

6.7.1 Factors affecting resource requirements

Number of agents/connections

The number of agents directly determines the number of connections to the Postgres database. Since every connection starts a new process, the connection count is a critical factor in the resource consumption of the entire Postgres deployment.

By default, most agents use one connection. If `enable_heartbeat_connection` is set to true, each agent uses two connections. Agents configured with `alert_threads` , `enable_smtp` , `enable_webhooks` , or `enable_snmp` set will open additional connections for those functionalities.

Since many of these connections are idle most of the time, you can safely set `max_connections` slightly higher than you would otherwise. However, it's crucial to remember this is still Postgres: once the connection count exceeds 100, connection overhead rapidly consumes resources. For this reason, for any large-scale PEM deployment, we strongly recommend deploying pgBouncer to manage connections efficiently.

Assuming you manage connections effectively either through a moderate number or with connection pooling in place, memory and CPU needs are then determined by the following factors.

Size of frequently accessed data - memory

Our memory sizing recommendations are designed to ensure the most frequently used data resides entirely in Postgres's shared buffers, minimizing disk I/O and maximizing performance. This high-priority, frequently accessed data includes:

- The `pemdata` schema, which stores the most recent data points collected by all probes.
- Various tables in the `pem` schema used for managing probe execution and alert dispatching.

The size of this critical data is primarily driven by the total number of probes and the volume of data each probe returns. In practical terms, this size is determined by the number of monitored database objects, particularly the more numerous objects like tables and indexes.

The recommended sizes that follow are based on the assumption that approximately 25% of RAM will be dedicated to shared buffers.

Rate of probe and alert execution - CPU and alert threads

PEM constantly performs two critical tasks: ingesting new probe data and evaluating that data against the configured alert thresholds. So PEM requires sufficient CPU time to execute all configured probes and alerts at their specified frequency. For instance, a setup with 100 alerts set to run once per minute demands a processing capacity of 100 alert evaluations per minute. Failure to meet this required execution rate results in delayed alerts or the possibility that alerts fail to trigger entirely.

The number of alert and probe executions scales with:

- The number of probes and alerts
- The configured frequency of the probes and alerts

Our recommended CPU numbers are calculated to ensure PEM can maintain the required execution rate for all probes and alerts using the default configuration. If you decide to customize your setup by enabling many additional alerts or probes or increasing the execution frequency, you may need to scale the CPU accordingly to prevent delays.

!!!Note Alert threads In larger deployments, you need to increase the total number of alert threads in the PEM agent configuration on the PEM hosts. This modification allows PEM to efficiently use more CPU time specifically for alert evaluation, preventing bottlenecks. You can change the number of alert threads by modifying the `alert_threads` setting of the PEM agent running on the PEM hosts.

Size of historical data - storage

PEM's storage needs are primarily determined by the volume of historical data it retains.

Three main factors drive this volume:

- Monitored object count _ The raw amount of data collected (typically dominated by numerous objects like tables and indexes).
- Data update frequency _ How often the probe data changes (PEM applies compression, so repeated, unchanged values don't consume extra space).
- Data retention period _ The configured length of time historical data is kept.

Our recommended storage sizes are based on PEM's default probe frequencies, standard retention periods, and an estimated data compression rate.

6.7.2 PEM t-shirt sizing

Important

For all the reasons explained above, the load on the PEM server critically depends on the nature of the estate it's monitoring. The size guide is a starting point. For large sizes, we strongly recommend that you start by adding a subset of your monitored servers to PEM and measuring resource usage. Don't be surprised if your eventual resource needs differ significantly from those we suggest.

Size	Description	Compo-nent	CPUs	RAM	Storage	IOPS	Other Notes
Nano	For testing PEM, 1–2 monitored servers, single user	—	1	2 GB	20 GB	<100	—

continues on next page

Table 2 – continued from previous page

Size	Description	Component	CPUs	RAM	Storage	IOPS	Other Notes
Tiny	For a single HA cluster, with a few web app users	—	2	4 GB	50 GB	<100	—
Small	For ≤50 servers, <5 concurrent web users; go higher if >50 k tables/indexes	—	4	8 GB	200 GB	300	—
Medium	For ≤100 servers, ~5 concurrent web users; go higher if >100 k tables/indexes	—	6	16 GB	300 GB	500	Use **pgBouncer** if heartbeat connections are enabled
Large	For ≤300 servers, ~10 concurrent users; go higher if >250 k tables/indexes	Backend	8	32 GB	1.5 TB	2000	~2 alert threads, **pgBouncer**
		Frontend	1–2	2–4 GB	20 GB	<100	—
X Large	For ≤600 servers, >10 concurrent users; go higher if >500 k tables/indexes	Backend	12	48 GB	3 TB	4000	~3 alert threads, **pgBouncer**
		Frontend	2	4 GB	20 GB	100	—
XX Large	For >600 servers, - recommend multiple PEM domains	—	—	—	—	—	Segment estate into multiple PEM deployments

6.8 High availability patterns for PEM deployment

If your PEM deployment must be highly available, you can deploy multiple instances of the PEM backend database and frontend web application in a high availability (HA) topology.

For detailed installation instructions for some typical configurations, see *Installing Postgres Enterprise Manager in HA Patterns* .

6.8.1 Running multiple instances of the PEM backend database

A highly available PEM backend database functions like any other standard HA Postgres cluster. The key requirements for the backend follow.

A primary

At any given time, only one instance of the PEM backend database can accept write connections. This instance is referred to as the *primary PEM backend database*, or simply the *primary*.

All other instances must function as replicas of the primary. You must implement a reliable method for promoting a replica in the event of a primary failure. This typically involves using a supported failover manager.

PEM supports the following failover solutions:

- EDB Failover Manager (EFM)
- Patroni

> **Note**
>
> EDB Postgres Distributed isn't supported as a PEM backend.

Connections must go to the primary

All connections from active PEM web application instances and running PEM agents must connect to the primary PEM backend database.

There are two general approaches to ensure this:

1. Use a *C1: Colocated with single endpoint* that always routes to the primary. You can implement this using methods like a virtual IP (VIP), a network load balancer, or a proxy that routes traffic to the current primary.
2. Configure clients to switch endpoints when the primary changes. This is commonly done using the multi-host connection strings, allowing clients to retry against alternate hosts when the primary is unreachable.

For more details, see *Load balancers, proxies, and VIPs* and *Reference architectures* .

6.8.2 Running multiple PEM web application instances

You can achieve high availability for the PEM web application by running multiple instances of the web application, all connected to the primary PEM backend, as described earlier. Unlike the backend, multiple frontend instances can run concurrently without issue.

To ensure consistent behavior across all instances:

- Configure all instances to store user preferences in the PEM backend database, not locally.
- For features that generate or store files, like dump/restore, use a shared file system accessible to all web instances.

6.8.3 Upgrading an HA PEM deployment

When running PEM in an HA configuration, it's critical to have a well-defined and tested upgrade procedure. This ensures minimal downtime and maintains the integrity of both the backend database and the web application during the upgrade process. For instructions for upgrading HA PEM, see *Upgrading an HA PEM installation* . Validate these instructions in your specific environment.

6.8.4 Load balancers, proxies, and VIPs

In the *Reference architectures* , the term Proxy/VIP refers to any component that presents a single endpoint for inbound connections and routes traffic to the current primary. You can implement this using a variety of tools and techniques, some of which are outlined here.

Using a virtual IP (VIP) managed by the failover manager

A virtual IP (VIP) is an IP address not tied to a specific physical network interface and can move between nodes in a subnet. Because VIPs are managed by the OS networking stack, they require no additional hardware or third-party software.

To route traffic to the primary using a VIP, configure your failover manager to assign the VIP to the new primary during a failover.

- EDB Failover Manager (EFM) supports VIPs natively. We recommend using EFM if you choose this method.
- Limitation: VIPs aren't suitable for multi-region or multi-subnet cloud environments.

Using a load balancer

If your environment supports load balancers (for example, AWS Elastic Load Balancer, on-prem F5, and so on), you can use them to route the traffic to the current primary. There are two common patterns:

- Use a failover manager that reconfigures the load balancer during the failover to route inbound connections to the new primary.
- Use a failover manager where the load balancer polls each node via an HTTP endpoint to determine which is the current primary. Only the primary responds positively (for example, HTTP code 200), meaning the load balancer switches all traffic to the primary within one poll interval. You can achieve this via the `/primary` or `/read-write` endpoints in Patroni, for example.

The second pattern is generally preferred, as it aligns with how load balancers are designed to operate and doesn't require administrative access to modify load balancer configurations during the failover.

> **Note**
>
> The reason we say "if your environment provides a load balancer" is that this solution needs to be properly implemented to avoid being a single point of failure. Load balancers must be highly available. This typically involves using DNS failover, elastic IPs, or other infrastructure-level solutions to avoid them becoming a single point of failure.

Adding a standalone instance of HAProxy or pgBouncer in front of your database isn't sufficient. EDB Support can't help you design or manage custom load balancer setups. The solution must be vetted, supported, and maintained by your organization's infrastructure or operations team.

Starting with PEM 10.1, both the PEM agent and the PEM web application support multi-host connection strings to connect to the backend database. With multi-host connection strings: - Clients (PEM agent or web application) try each listed host until one accepts write connections. - On failover, connections are dropped and the client begins the search again. - If the first host in the string isn't the primary, a failed connection attempt occurs. This failure increases the time it takes to establish a connection to the primary and increases resource usage on the monitored server and the PEM backend servers.

6.8.5 Reference architectures

All the architectures that follow show three nodes with PEM backend databases. However, this approach also works with only two nodes, providing the failover manager is okay with even node numbers. (If not, you can use a witness node in place of a full third node). More than three nodes also works fine, but that configuration may not yield significant benefit for most deployments.

C1: Colocated with single endpoint

A minimal setup where the PEM web application and backend database both run on the same host.

Pros:

- Transparent connections to the same endpoint for both web application users and monitoring agents.
- Simplified web application configuration and upgrades, as all instances connect to the loopback interface.
- Suitable for smaller deployments.

Cons:

- Shared resources may become a bottleneck.
- No resilience to failure of the web application. Failover occurs if the database fails unless you customize the failover conditions.

- Web applications running on the replicas aren't useable as the web application will refuse to connect to Postgres servers that aren't writeable.

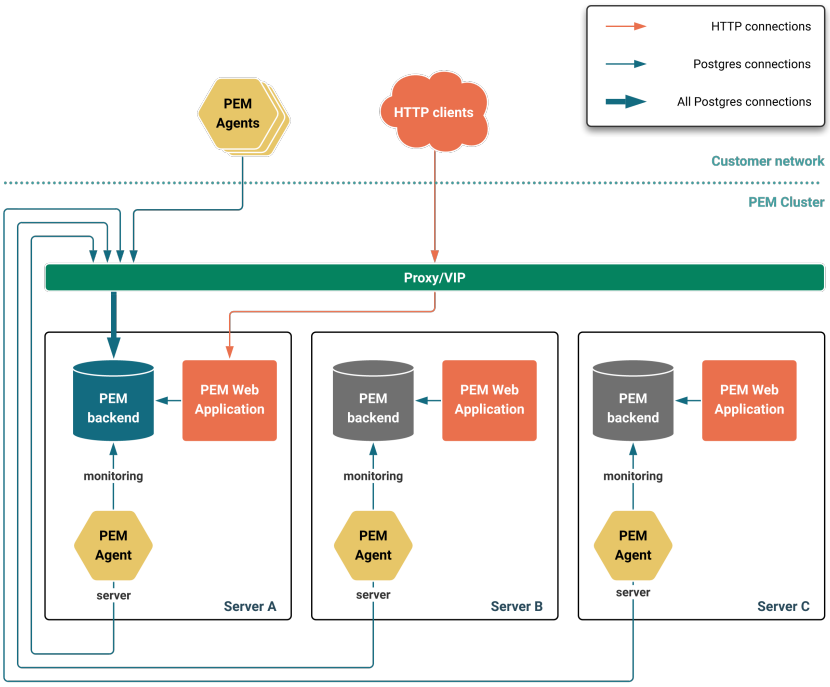


Fig. 3: reference arch

S1: Separated with single endpoint for database

A setup where the PEM web application and backend database both run on separate hosts.

Pros:

- Transparent connections for monitoring agents, as they always connect to the same endpoint.
- Web application and database components are independently scalable and redundant.
- Suitable for larger deployments.

Cons:

- Users must manually choose a web application instance unless you use a frontend load balancer.

CM: Colocated with multi-host connection strings

A single-endpoint setup is replaced by clients using multi-host connection strings to ensure traffic is routed to the primary. Web application and backend database are colocated.

Pros:

- No need to manage a load balancer or VIP for the database server.
- Redundant web applications.
- Smaller footprint than S1 and SM.

Cons:

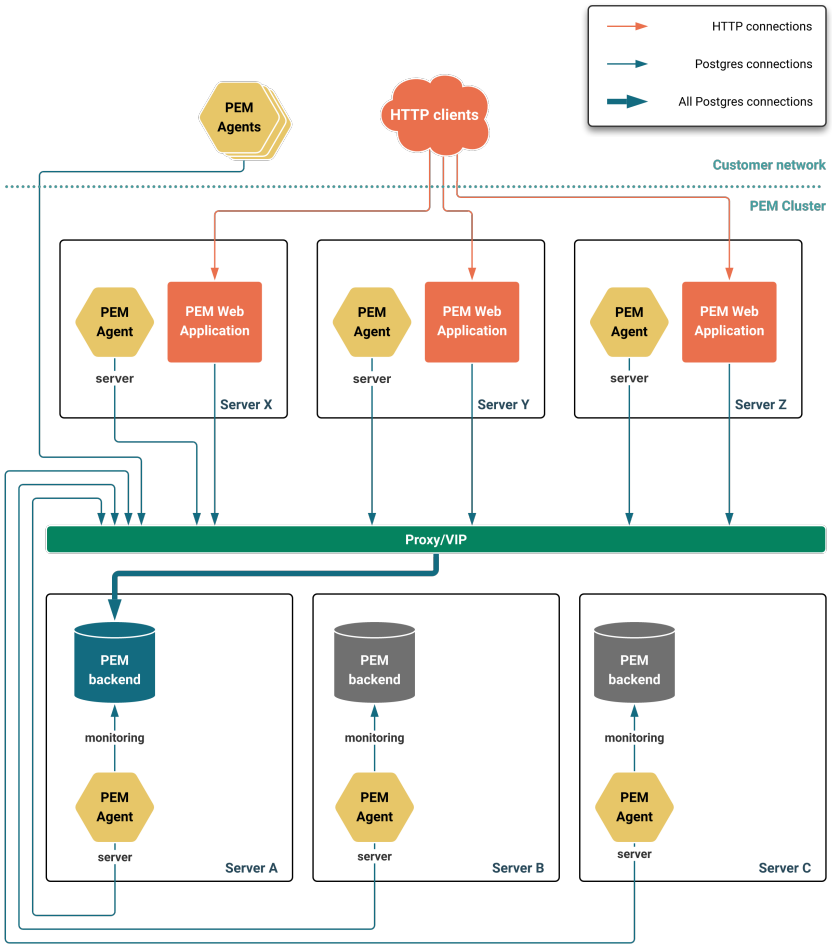


Fig. 4: reference arch

- Hard to reconfigure as every monitoring agent and web application must be configured with the connection details of all backends databases.
- It can be confusing to have multiple instances of the web application running. Users must manually choose a web application instance unless you use a frontend load balancer.
- Multi-host connections are less efficient than routing through a single endpoint.

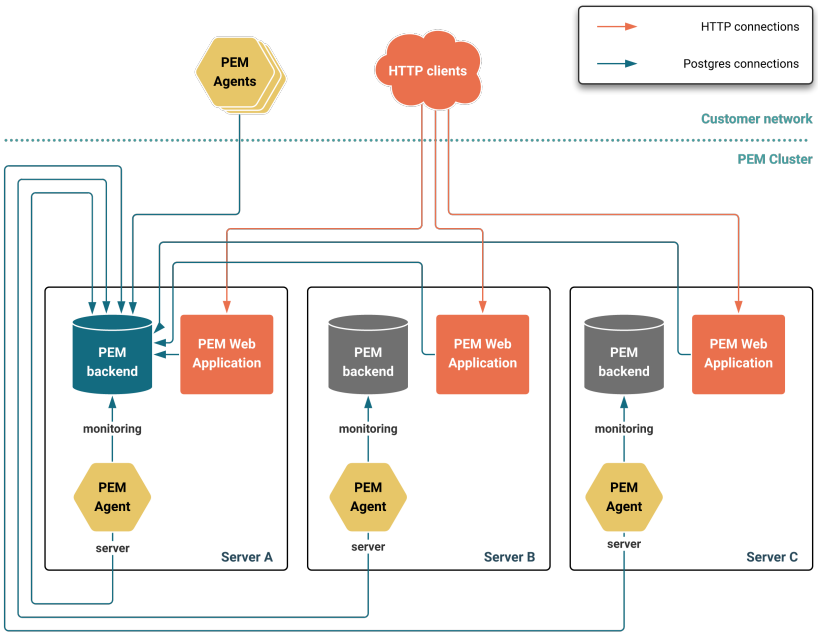


Fig. 5: reference arch

SM: Separated with multi-host connection strings

This setup removes the single endpoint but separates the web application and backend database for improved scalability.

Pros:

- No need to manage a load balancer or VIP for the database server.
- Redundant web application and backend database components.
- Suitable for larger deployments for improved scalability.

Cons:

- Hard to reconfigure as every monitoring agent and web application must be configured with the connection details of all backend databases.
- It can be confusing to have multiple instances of the web application running. Users must manually choose a web application instance unless you're using a frontend load balancer.
- Multi-host connections are less efficient than routing through a single endpoint

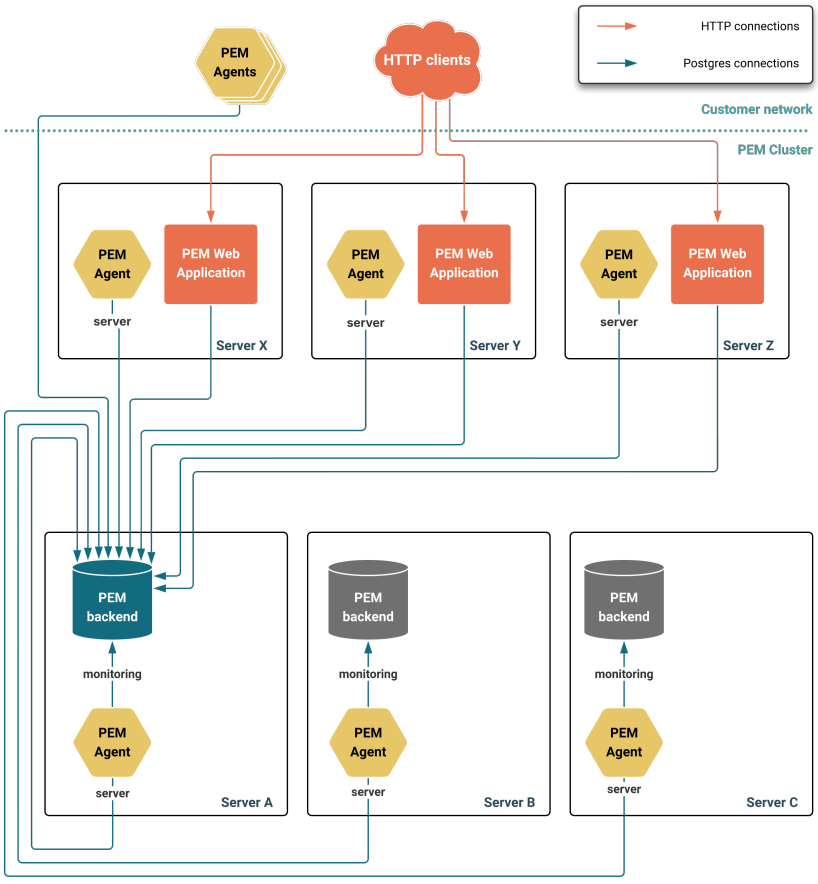


Fig. 6: reference arch

CHAPTER 7

Installing Postgres Enterprise Manager server

Select a link to access the applicable installation instructions:

7.1 Linux x86-64 (amd64)

7.1.1 Red Hat Enterprise Linux (RHEL) and derivatives

- *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 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*

7.1.2 SUSE Linux Enterprise (SLES)

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

7.1.3 Debian and derivatives

- *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*

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

7.2.1 Red Hat Enterprise Linux (RHEL) and derivatives

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

7.2.2 SUSE Linux Enterprise (SLES)

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

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

7.3.1 Red Hat Enterprise Linux (RHEL) and derivatives

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

7.3.2 Debian and derivatives

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

7.4 Windows

- *Windows Server 2019*

7.4.1 Dependencies of the PEM Server and Agent on Linux

The PEM Server and Agent packages for Linux have dependencies on various system libraries. These dependencies are detailed below for reference.

> **Note**
>
> A PEM Agent is always installed alongside PEM Server, so all dependencies must be present on hosts where PEM Server (either the database or the web application) is installed.

Typically, PEM is built against the latest version of each dependency available from the vendor repository for a given platform and architecture. In some cases, PEM requires a newer version of a library than is available in the vendor repository. In these cases a newer version of the package, prefixed with `edb-` is sourced from EDB's repositories.

> **Note**
>
> This information is provided for reference. Packages from vendor repositories are not supported or patched by EDB. Refer to your operating system documentation or support provider for details of these packages.

Because these dependencies are updated frequently, the tables below are valid only for the latest patch release of PEM.

Python 3 is required for PEM Server. If Apache HTTPD is used as the web server, then the mod_wsgi module is also required.

Platform	Architecture	Python/mod_wsgi package	Python version	Python path
	ppc64le	*edb-python312/edb-python312-mo*	3.12	*/usr/libexec/edb-python312/bin/python3*
	s390x	*edb-python312/edb-python312-mo*	3.12	*/usr/libexec/edb-python312/bin/python3*
RHEL 8	x86_64	*edb-python312/edb-python312-mo*	3.12	*/usr/libexec/edb-python312/bin/python3*
RHEL 9	x86_64	*edb-python312/edb-python312-mo*	3.12	*/usr/libexec/edb-python312/bin/python3*
	ppc64le	*edb-python312/edb-python312-mo*	3.12	*/usr/libexec/edb-python312/bin/python3*
	s390x	*edb-python312/edb-python312-mo*	3.12	*/usr/libexec/edb-python312/bin/python3*
	arm64	*edb-python312/edb-python312-mo*	3.12	*/usr/libexec/edb-python312/bin/python3*
SLES 15	x86_64	*edb-python312/edb-python312-mo*	3.12	*/usr/libexec/edb-python312/bin/python3*
	ppc64le	*edb-python312/edb-python312-mo*	3.12	*/usr/libexec/edb-python312/bin/python3*
	s390x	*edb-python312/edb-python312-mo*	3.12	*/usr/libexec/edb-python312/bin/python3*
Ubuntu 22	amd64	*edb-python312/edb-python312-mo*	3.12	*/usr/libexec/edb-python312/bin/python3*
Ubuntu 24	amd64	*python312/liba-pache2-mod-wsgi-py3*	3.12	*/usr/bin/python3*
Debian 11	amd64	*edb-python312/edb-python312-mo*	3.12	*/usr/libexec/edb-python312/bin/python3*
Debian 12	amd64/arm	*edb-python312/edb-python312-mo*	3.12	*/usr/libexec/edb-python312/bin/python3*

OpenSSL

The PEM Server and Agent require OpenSSL.

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-3.0.7*
	ppc64le	*openssl-3.0.7*
	s390x	*openssl-3.0.7*
	arm64	*openssl-3.0.7*
SLES 15	x86_64	*openssl-1.1.1l*
	ppc64le	*openssl-1.1.1l*
	s390x	*openssl-1.1.1l*
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-3.0.11*

Libcurl

The PEM Agent requires libcurl.

Platform	Architecture	package-version
RHEL 8	x86_64	*curl-7.61.1*
	ppc64le	*curl-7.61.1*
	s390x	*curl-7.61.1*

continues on next page

Table 3 – continued from previous page

Platform	Architecture	package-version
RHEL 9	x86_64	*curl-7.76.1*
	ppc64le	*curl-7.76.1*
	s390x	*curl-7.76.1*
	arm64	*curl-7.76.1*
SLES 15	x86_64	*curl-8.0.1*
	ppc64le	*curl-8.0.1*
	s390x	*curl-8.0.1*
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++

The PEM Agent requires SNMP++. All platforms use `edb-snmp++ 3.6.3`.

Boost libraries

The PEM Agent requires the Boost libraries.

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*
SLES 15	x86_64	*libboost_regex1_66_1-1.66.0*
	ppc64le	*libboost_regex1_66_1-1.66.0*
	s390x	*libboost_regex1_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*

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

Operating system-specific install instructions are described in the corresponding documentation:

Red Hat Enterprise Linux (RHEL) and derivatives

- *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 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 and derivatives

- *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 9 or OL 9 x86_64

You can install PEM on a single server, or you can install the web application server and the backend database on two separate servers. You must prepare your servers for PEM installation.

After fulfilling the prerequisites and completing the installation procedure described in the following steps, you must *Configure the web server (NGINX)* PEM. If you're using two servers, install and configure PEM on both servers.

Prerequisites

Before you begin the installation process:

1. Install a supported Postgres instance for PEM to use as a backend database.

You can install this instance on the same server to be used for the PEM web application or on a separate server. You can also use an existing Postgres instance if it is configured as detailed in the next steps.

2. Configure authentication on the Postgres backend database by updating the `pg_hba.conf` file.

Make the following changes manually, prior to configuration. (Additional changes are necessary during *Modify the SELinux configuration* .)

- To create the relations required for PEM, the PEM configuration script connects to the Postgres backend database as a superuser of your choice using password authentication. This requires you to permit your chosen superuser to authenticate using a password. This user must be able to connect from any location where you run the configuration script. In practice, this means the server where the backend database is located and the server where the PEM web application is to be installed, if they're different.
- To allow the chosen superuser to connect using password authentication, add a line to `pg_hba.conf` that allows `host` connections using `md5` or `scram-sha-256` authentication, such as `host all superusername 127.0.0.1/32 scram-sha-256` .

> **Note**
>
> If you're using EDB Postgres Advanced Server, see Modifying the pg_hba.conf file .

If you're using PostgreSQL, see Client Authentication . .. ::
2_blank_1_3. Verify that the *sslutils* extension is installed on your Postgres server.

If you're using PostgreSQL or EDB Postgres Extended Server on RHEL/AlmaLinux/Rocky Linux or SLES, you also need to install the `hstore contrib` module.

- If you're using EDB Postgres Advanced Server, you can install the `sslutils` extension as follows, where `<x>` is the EDB Postgres Advanced server version.

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

- If you're using PostgreSQL, you can install the `sslutils` and, if required, `hstore` modules as follows, where `<x>` is the PostgreSQL version.

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

- If you're using EDB Postgres Extended Server, you can install the `sslutils` and, if required, `hstore` modules as follows, where `<x>` is the EDB Postgres Extended Server version.

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

4. If you're using a firewall, allow access to port 8443 on the server where the PEM web application will be located:

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

firewall-cmd --reload
```

5. Make sure the components Postgres Enterprise Manager depends on are up to date on all servers. You can do this by updating the whole system using your package manager as shown below. If you prefer to update individual packages, a full list of dependencies is provided in *Dependencies of the PEM Server and Agent on Linux* .

```
sudo dnf upgrade
```

Install the package

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

Initial configuration

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

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

For more details, see *Configuring the PEM server on Linux* .

Note

- The operating system user pem is created while installing the PEM server. The PEM application data and the session is saved to this user's home directory.

Supported locales

Currently, the Postgres Enterprise Manager server and web interface support a locale of `English(US) en_US` and use of a period (.) as a language separator character. Using an alternate locale or a separator character other than a period might cause errors.

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

You can install PEM on a single server, or you can install the web application server and the backend database on two separate servers. You must prepare your servers for PEM installation.

After fulfilling the prerequisites and completing the installation procedure described in the following steps, you must *Configure the web server (NGINX)* PEM. If you're using two servers, install and configure PEM on both servers.

Prerequisites

Before you begin the installation process:

1. Install a supported Postgres instance for PEM to use as a backend database.

You can install this instance on the same server to be used for the PEM web application or on a separate server. You can also use an existing Postgres instance if it is configured as detailed in the next steps.

2. Configure authentication on the Postgres backend database by updating the `pg_hba.conf` file.

Make the following changes manually, prior to configuration. (Additional changes are necessary during *Modify the SELinux configuration* .)

- To create the relations required for PEM, the PEM configuration script connects to the Postgres backend database as a superuser of your choice using password authentication. This requires you to permit your chosen superuser to authenticate using a password. This user must be able to connect from any location where you run the configuration script. In practice, this means the server where the backend database is located and the server where the PEM web application is to be installed, if they're different.
- To allow the chosen superuser to connect using password authentication, add a line to `pg_hba.conf` that allows `host` connections using `md5` or `scram-sha-256` authentication, such as `host all superusername 127.0.0.1/32 scram-sha-256` .

> **Note**
>
> If you're using EDB Postgres Advanced Server, see Modifying the pg_hba.conf file .

If you're using PostgreSQL, see Client Authentication . .. ::
2_blank_1_3. Verify that the *sslutils* extension is installed on your Postgres server.

If you're using PostgreSQL or EDB Postgres Extended Server on RHEL/AlmaLinux/Rocky Linux or SLES, you also need to install the `hstore contrib` module.

- If you're using EDB Postgres Advanced Server, you can install the `sslutils` extension as follows, where `<x>` is the EDB Postgres Advanced server version.

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

- If you're using PostgreSQL, you can install the `sslutils` and, if required, `hstore` modules as follows, where `<x>` is the PostgreSQL version.

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

- If you're using EDB Postgres Extended Server, you can install the `sslutils` and, if required, `hstore` modules as follows, where `<x>` is the EDB Postgres Extended Server version.

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

4. If you're using a firewall, allow access to port 8443 on the server where the PEM web application will be located:

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

firewall-cmd --reload
```

5. Make sure the components Postgres Enterprise Manager depends on are up to date on all servers. You can do this by updating the whole system using your package manager as shown below. If you prefer to update individual packages, a full list of dependencies is provided in *Dependencies of the PEM Server and Agent on Linux* .

```
sudo dnf upgrade
```

Install the package

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

Initial configuration

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

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

For more details, see *Configuring the PEM server on Linux* .

Note

- The operating system user pem is created while installing the PEM server. The PEM application data and the session is saved to this user's home directory.

Supported locales

Currently, the Postgres Enterprise Manager server and web interface support a locale of `English(US) en_US` and use of a period (.) as a language separator character. Using an alternate locale or a separator character other than a period might cause errors.

Installing Postgres Enterprise Manager server on AlmaLinux 9 or Rocky

You can install PEM on a single server, or you can install the web application server and the backend database on two separate servers. You must prepare your servers for PEM installation.

After fulfilling the prerequisites and completing the installation procedure described in the following steps, you must *Configure the web server (NGINX)* PEM. If you're using two servers, install and configure PEM on both servers.

Prerequisites

Before you begin the installation process:

1. Install a supported Postgres instance for PEM to use as a backend database.

You can install this instance on the same server to be used for the PEM web application or on a separate server. You can also use an existing Postgres instance if it is configured as detailed in the next steps.

2. Configure authentication on the Postgres backend database by updating the `pg_hba.conf` file.

Make the following changes manually, prior to configuration. (Additional changes are necessary during *Modify the SELinux configuration* .)

- To create the relations required for PEM, the PEM configuration script connects to the Postgres backend database as a superuser of your choice using password authentication. This requires you to permit your chosen superuser to authenticate using a password. This user must be able to connect from any location where you run the configuration script. In practice, this means the server where the backend database is located and the server where the PEM web application is to be installed, if they're different.
- To allow the chosen superuser to connect using password authentication, add a line to `pg_hba.conf` that allows `host` connections using `md5` or `scram-sha-256` authentication, such as `host all superusername 127.0.0.1/32 scram-sha-256` .

> **Note**
>
> If you're using EDB Postgres Advanced Server, see Modifying the pg_hba.conf file .

If you're using PostgreSQL, see Client Authentication . .. ::

2_blank_1_3. Verify that the *sslutils* extension is installed on your Postgres server.

If you're using PostgreSQL or EDB Postgres Extended Server on RHEL/AlmaLinux/Rocky Linux or SLES, you also need to install the `hstore contrib` module.

- If you're using EDB Postgres Advanced Server, you can install the `sslutils` extension as follows, where `<x>` is the EDB Postgres Advanced server version.

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

- If you're using PostgreSQL, you can install the `sslutils` and, if required, `hstore` modules as follows, where `<x>` is the PostgreSQL version.

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

- If you're using EDB Postgres Extended Server, you can install the `sslutils` and, if required, `hstore` modules as follows, where `<x>` is the EDB Postgres Extended Server version.

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

4. If you're using a firewall, allow access to port 8443 on the server where the PEM web application will be located:

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

firewall-cmd --reload
```

5. Make sure the components Postgres Enterprise Manager depends on are up to date on all servers. You can do this by updating the whole system using your package manager as shown below. If you prefer to update individual packages, a full list of dependencies is provided in *Dependencies of the PEM Server and Agent on Linux* .

```
sudo dnf upgrade
```

Install the package

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

Initial configuration

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

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

For more details, see *Configuring the PEM server on Linux* .

Note

- The operating system user pem is created while installing the PEM server. The PEM application data and the session is saved to this user's home directory.

Supported locales

Currently, the Postgres Enterprise Manager server and web interface support a locale of `English(US) en_US` and use of a period (.) as a language separator character. Using an alternate locale or a separator character other than a period might cause errors.

Installing Postgres Enterprise Manager server on AlmaLinux 8 or Rocky

You can install PEM on a single server, or you can install the web application server and the backend database on two separate servers. You must prepare your servers for PEM installation.

After fulfilling the prerequisites and completing the installation procedure described in the following steps, you must *Configure the web server (NGINX)* PEM. If you're using two servers, install and configure PEM on both servers.

Prerequisites

Before you begin the installation process:

1. Install a supported Postgres instance for PEM to use as a backend database.

You can install this instance on the same server to be used for the PEM web application or on a separate server. You can also use an existing Postgres instance if it is configured as detailed in the next steps.

2. Configure authentication on the Postgres backend database by updating the `pg_hba.conf` file.

Make the following changes manually, prior to configuration. (Additional changes are necessary during *Modify the SELinux configuration* .)

- To create the relations required for PEM, the PEM configuration script connects to the Postgres backend database as a superuser of your choice using password authentication. This requires you to permit your chosen superuser to authenticate using a password. This user must be able to connect from any location where you run the configuration script. In practice, this means the server where the backend database is located and the server where the PEM web application is to be installed, if they're different.
- To allow the chosen superuser to connect using password authentication, add a line to `pg_hba.conf` that allows `host` connections using `md5` or `scram-sha-256` authentication, such as `host all superusername 127.0.0.1/32 scram-sha-256` .

Note

If you're using EDB Postgres Advanced Server, see Modifying the pg_hba.conf file .

If you're using PostgreSQL, see Client Authentication . .. ::

2_blank_1_3. Verify that the *sslutils* extension is installed on your Postgres server.

If you're using PostgreSQL or EDB Postgres Extended Server on RHEL/AlmaLinux/Rocky Linux or SLES, you also need to install the `hstore contrib` module.

- If you're using EDB Postgres Advanced Server, you can install the `sslutils` extension as follows, where `<x>` is the EDB Postgres Advanced server version.

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

- If you're using PostgreSQL, you can install the `sslutils` and, if required, `hstore` modules as follows, where `<x>` is the PostgreSQL version.

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

- If you're using EDB Postgres Extended Server, you can install the `sslutils` and, if required, `hstore` modules as follows, where `<x>` is the EDB Postgres Extended Server version.

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

4. If you're using a firewall, allow access to port 8443 on the server where the PEM web application will be located:

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

firewall-cmd --reload
```

5. Make sure the components Postgres Enterprise Manager depends on are up to date on all servers. You can do this by updating the whole system using your package manager as shown below. If you prefer to update individual packages, a full list of dependencies is provided in *Dependencies of the PEM Server and Agent on Linux* .

```
sudo dnf upgrade
```

Install the package

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

Initial configuration

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

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

For more details, see *Configuring the PEM server on Linux* .

Note

- The operating system user pem is created while installing the PEM server. The PEM application data and the session is saved to this user's home directory.

Supported locales

Currently, the Postgres Enterprise Manager server and web interface support a locale of `English(US) en_US` and use of a period (.) as a language separator character. Using an alternate locale or a separator character other than a period might cause errors.

Installing Postgres Enterprise Manager server on SLES 15 x86_64

> **Note**
>
> Postgres Enterprise Manager 8.3 and later is supported on SLES.

Prerequisites

Before you begin the installation process:

1. Install a supported Postgres instance for PEM to use as a backend database.

You can install this instance on the same server to be used for the PEM web application or on a separate server. You can also use an existing Postgres instance if it is configured as detailed in the next steps.

2. Configure authentication on the Postgres backend database by updating the `pg_hba.conf` file.

Make the following changes manually, prior to configuration. (Additional changes are necessary during *Modify the SELinux configuration* .)

- To create the relations required for PEM, the PEM configuration script connects to the Postgres backend database as a superuser of your choice using password authentication. This requires you to permit your chosen superuser to authenticate using a password. This user must be able to connect from any location where you run the configuration script. In practice, this means the server where the backend database is located and the server where the PEM web application is to be installed, if they're different.
- To allow the chosen superuser to connect using password authentication, add a line to `pg_hba.conf` that allows `host` connections using `md5` or `scram-sha-256` authentication, such as `host all superusername 127.0.0.1/32 scram-sha-256` .

> **Note**
>
> If you're using EDB Postgres Advanced Server, see Modifying the pg_hba.conf file .

If you're using PostgreSQL, see Client Authentication . .. ::
2_blank_1_3. Verify that the *sslutils* extension is installed on your Postgres server.

If you're using PostgreSQL or EDB Postgres Extended Server on RHEL/AlmaLinux/Rocky Linux or SLES, you also need to install the `hstore contrib` module.

- If you're using EDB Postgres Advanced Server, you can install the `sslutils` extension as follows, where `<x>` is the EDB Postgres Advanced server version.

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

- If you're using PostgreSQL, you can install the `sslutils` and, if required, `hstore` modules as follows, where `<x>` is the PostgreSQL version.

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

- If you're using EDB Postgres Extended Server, you can install the `sslutils` and, if required, `hstore` modules as follows, where `<x>` is the EDB Postgres Extended Server version.

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

4. If you're using a firewall, allow access to port 8443 on the server where the PEM web application will be located:

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

firewall-cmd --reload
```

5. Make sure the components Postgres Enterprise Manager depends on are up to date on all servers. You can do this by updating the whole system using your package manager as shown below. If you prefer to update individual packages, a full list of dependencies is provided in *Dependencies of the PEM Server and Agent on Linux* .

```
sudo zypper update
```

Install the package

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

Initial configuration

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

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

For more details, see *Configuring the PEM server on Linux* .

Note

- The operating system user pem is created while installing the PEM server. The PEM application data and the session is saved to this user's home directory.

Supported locales

Currently, the Postgres Enterprise Manager server and web interface support a locale of `English(US) en_US` and use of a period (.) as a language separator character. Using an alternate locale or a separator character other than a period might cause errors.

Installing Postgres Enterprise Manager server on Ubuntu 24.04 x86_64

You can install PEM on a single server, or you can install the web application server and the backend database on two separate servers. You must prepare your servers for PEM installation.

After fulfilling the prerequisites and completing the installation procedure described in the following steps, you must *Configure the web server (NGINX)* PEM. If you're using two servers, install and configure PEM on both servers.

Prerequisites

Before you begin the installation process:

1. Install a supported Postgres instance for PEM to use as a backend database.

You can install this instance on the same server to be used for the PEM web application or on a separate server. You can also use an existing Postgres instance if it is configured as detailed in the next steps.

2. Configure authentication on the Postgres backend database by updating the `pg_hba.conf` file.

Make the following changes manually, prior to configuration. (Additional changes are necessary during *Modify the SELinux configuration* .)

- To create the relations required for PEM, the PEM configuration script connects to the Postgres backend database as a superuser of your choice using password authentication. This requires you to permit your chosen superuser to authenticate using a password. This user must be able to connect from any location where you run the configuration script. In practice, this means the server where the backend database is located and the server where the PEM web application is to be installed, if they're different.
- To allow the chosen superuser to connect using password authentication, add a line to `pg_hba.conf` that allows `host` connections using `md5` or `scram-sha-256` authentication, such as `host all superusername 127.0.0.1/32 scram-sha-256` .

> **Note**
>
> If you're using EDB Postgres Advanced Server, see Modifying the pg_hba.conf file .

If you're using PostgreSQL, see Client Authentication . .. ::
2_blank_1_3. Verify that the *sslutils* extension is installed on your Postgres server.

If you're using PostgreSQL or EDB Postgres Extended Server on RHEL/AlmaLinux/Rocky Linux or SLES, you also need to install the `hstore contrib` module.

- If you're using EDB Postgres Advanced Server, you can install the `sslutils` extension as follows, where `<x>` is the EDB Postgres Advanced server version.

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

- If you're using PostgreSQL, you can install the `sslutils` and, if required, `hstore` modules as follows, where `<x>` is the PostgreSQL version.

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

- If you're using EDB Postgres Extended Server, you can install the `sslutils` and, if required, `hstore` modules as follows, where `<x>` is the EDB Postgres Extended Server version.

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

> **Note**
>
> Ubuntu 20 changed the requirements for accepting certificates.

```
- If you want to install the PEM agent on a machine with an old version of sslutils,␣
↪then you must upgrade sslutils to 1.3. Version 1.3 has a 4096-bit RSA key and␣
```

(continues on next page)

(continued from previous page)

```
↪sha256 signature algorithm support added to it. - If you don't upgrade sslutils to␣
↪1.3, then PEM agent might fail to connect to the PEM backend database server, and␣
↪it might log the error "ca md too weak."
```

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

5. Make sure the components Postgres Enterprise Manager depends on are up to date on all servers. You can do this by updating the whole system using your package manager as shown below. If you prefer to update individual packages, a full list of dependencies is provided in *Dependencies of the PEM Server and Agent on Linux* .

```
sudo apt-get update
```

Install the package

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

Initial configuration

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

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

For more details, see *Configuring the PEM server on Linux* .

Note

- The operating system user pem is created while installing the PEM server. The PEM application data and the session is saved to this user's home directory.

Supported locales

Currently, the Postgres Enterprise Manager server and web interface support a locale of `English(US) en_US` and use of a period (.) as a language separator character. Using an alternate locale or a separator character other than a period might cause errors.

Installing Postgres Enterprise Manager server on Ubuntu 22.04 x86_64

You can install PEM on a single server, or you can install the web application server and the backend database on two separate servers. You must prepare your servers for PEM installation.

After fulfilling the prerequisites and completing the installation procedure described in the following steps, you must *Configure the web server (NGINX)* PEM. If you're using two servers, install and configure PEM on both servers.

Prerequisites

Before you begin the installation process:

1. Install a supported Postgres instance for PEM to use as a backend database.

You can install this instance on the same server to be used for the PEM web application or on a separate server. You can also use an existing Postgres instance if it is configured as detailed in the next steps.

2. Configure authentication on the Postgres backend database by updating the `pg_hba.conf` file.

Make the following changes manually, prior to configuration. (Additional changes are necessary during *Modify the SELinux configuration* .)

- To create the relations required for PEM, the PEM configuration script connects to the Postgres backend database as a superuser of your choice using password authentication. This requires you to permit your chosen superuser to authenticate using a password. This user must be able to connect from any location where you run the configuration script. In practice, this means the server where the backend database is located and the server where the PEM web application is to be installed, if they're different.
- To allow the chosen superuser to connect using password authentication, add a line to `pg_hba.conf` that allows `host` connections using `md5` or `scram-sha-256` authentication, such as `host all superusername 127.0.0.1/32 scram-sha-256` .

Note

If you're using EDB Postgres Advanced Server, see Modifying the pg_hba.conf file .

If you're using PostgreSQL, see Client Authentication . .. ::

2_blank_1_3. Verify that the *sslutils* extension is installed on your Postgres server.

If you're using PostgreSQL or EDB Postgres Extended Server on RHEL/AlmaLinux/Rocky Linux or SLES, you also need to install the `hstore contrib` module.

- If you're using EDB Postgres Advanced Server, you can install the `sslutils` extension as follows, where `<x>` is the EDB Postgres Advanced server version.

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

- If you're using PostgreSQL, you can install the `sslutils` and, if required, `hstore` modules as follows, where `<x>` is the PostgreSQL version.

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

- If you're using EDB Postgres Extended Server, you can install the `sslutils` and, if required, `hstore` modules as follows, where `<x>` is the EDB Postgres Extended Server version.

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

4. If you're using a firewall, allow access to port 8443 on the server where the PEM web application will be located:

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

5. Make sure the components Postgres Enterprise Manager depends on are up to date on all servers. You can do this by updating the whole system using your package manager as shown below. If you prefer to update individual packages, a full list of dependencies is provided in *Dependencies of the PEM Server and Agent on Linux* .

```
sudo apt-get update
```

Install the package

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

Initial configuration

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

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

For more details, see *Configuring the PEM server on Linux* .

Note

- The operating system user pem is created while installing the PEM server. The PEM application data and the session is saved to this user's home directory.

Supported locales

Currently, the Postgres Enterprise Manager server and web interface support a locale of `English(US) en_US` and use of a period (.) as a language separator character. Using an alternate locale or a separator character other than a period might cause errors.

Installing Postgres Enterprise Manager server on Debian 12 x86_64

You can install PEM on a single server, or you can install the web application server and the backend database on two separate servers. You must prepare your servers for PEM installation.

After fulfilling the prerequisites and completing the installation procedure described in the following steps, you must *Configure the web server (NGINX)* PEM. If you're using two servers, install and configure PEM on both servers.

Prerequisites

Before you begin the installation process:

1. Install a supported Postgres instance for PEM to use as a backend database.

You can install this instance on the same server to be used for the PEM web application or on a separate server. You can also use an existing Postgres instance if it is configured as detailed in the next steps.

2. Configure authentication on the Postgres backend database by updating the `pg_hba.conf` file.

Make the following changes manually, prior to configuration. (Additional changes are necessary during *Modify the SELinux configuration* .)

- To create the relations required for PEM, the PEM configuration script connects to the Postgres backend database as a superuser of your choice using password authentication. This requires you to permit your chosen superuser to authenticate using a password. This user must be able to connect from any location where you run the configuration script. In practice, this means the server where the backend database is located and the server where the PEM web application is to be installed, if they're different.
- To allow the chosen superuser to connect using password authentication, add a line to `pg_hba.conf` that allows `host` connections using `md5` or `scram-sha-256` authentication, such as `host all superusername 127.0.0.1/32 scram-sha-256` .

Note

If you're using EDB Postgres Advanced Server, see Modifying the pg_hba.conf file .

If you're using PostgreSQL, see Client Authentication . .. ::

2_blank_1_3. Verify that the *sslutils* extension is installed on your Postgres server.

If you're using PostgreSQL or EDB Postgres Extended Server on RHEL/AlmaLinux/Rocky Linux or SLES, you also need to install the `hstore contrib` module.

- If you're using EDB Postgres Advanced Server, you can install the `sslutils` extension as follows, where `<x>` is the EDB Postgres Advanced server version.

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

- If you're using PostgreSQL, you can install the `sslutils` and, if required, `hstore` modules as follows, where `<x>` is the PostgreSQL version.

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

- If you're using EDB Postgres Extended Server, you can install the `sslutils` and, if required, `hstore` modules as follows, where `<x>` is the EDB Postgres Extended Server version.

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

4. If you're using a firewall, allow access to port 8443 on the server where the PEM web application will be located:
5. Make sure the components Postgres Enterprise Manager depends on are up to date on all servers. You can do this by updating the whole system using your package manager as shown below. If you prefer to update individual packages, a full list of dependencies is provided in *Dependencies of the PEM Server and Agent on Linux* .

```
sudo apt-get upgrade
```

Install the package

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

Initial configuration

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

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

For more details, see *Configuring the PEM server on Linux* .

Note

- The operating system user pem is created while installing the PEM server. The PEM application data and the session is saved to this user's home directory.

Supported locales

Currently, the Postgres Enterprise Manager server and web interface support a locale of `English(US) en_US` and use of a period (.) as a language separator character. Using an alternate locale or a separator character other than a period might cause errors.

Installing Postgres Enterprise Manager server on Debian 11 x86_64

You can install PEM on a single server, or you can install the web application server and the backend database on two separate servers. You must prepare your servers for PEM installation.

After fulfilling the prerequisites and completing the installation procedure described in the following steps, you must *Configure the web server (NGINX)* PEM. If you're using two servers, install and configure PEM on both servers.

Prerequisites

Before you begin the installation process:

1. Install a supported Postgres instance for PEM to use as a backend database.

You can install this instance on the same server to be used for the PEM web application or on a separate server. You can also use an existing Postgres instance if it is configured as detailed in the next steps.

2. Configure authentication on the Postgres backend database by updating the `pg_hba.conf` file.

Make the following changes manually, prior to configuration. (Additional changes are necessary during *Modify the SELinux configuration* .)

- To create the relations required for PEM, the PEM configuration script connects to the Postgres backend database as a superuser of your choice using password authentication. This requires you to permit your chosen superuser to authenticate using a password. This user must be able to connect from any location where you run the configuration script. In practice, this means the server where the backend database is located and the server where the PEM web application is to be installed, if they're different.
- To allow the chosen superuser to connect using password authentication, add a line to `pg_hba.conf` that allows `host` connections using `md5` or `scram-sha-256` authentication, such as `host all superusername 127.0.0.1/32 scram-sha-256` .

> **Note**
>
> If you're using EDB Postgres Advanced Server, see Modifying the pg_hba.conf file .

If you're using PostgreSQL, see Client Authentication . .. ::
2_blank_1_3. Verify that the *sslutils* extension is installed on your Postgres server.

If you're using PostgreSQL or EDB Postgres Extended Server on RHEL/AlmaLinux/Rocky Linux or SLES, you also need to install the `hstore contrib` module.

- If you're using EDB Postgres Advanced Server, you can install the `sslutils` extension as follows, where `<x>` is the EDB Postgres Advanced server version.

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

- If you're using PostgreSQL, you can install the `sslutils` and, if required, `hstore` modules as follows, where `<x>` is the PostgreSQL version.

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

- If you're using EDB Postgres Extended Server, you can install the `sslutils` and, if required, `hstore` modules as follows, where `<x>` is the EDB Postgres Extended Server version.

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

4. If you're using a firewall, allow access to port 8443 on the server where the PEM web application will be located:

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

5. Make sure the components Postgres Enterprise Manager depends on are up to date on all servers. You can do this by updating the whole system using your package manager as shown below. If you prefer to update individual packages, a full list of dependencies is provided in *Dependencies of the PEM Server and Agent on Linux* .

```
sudo apt-get update
```

Install the package

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

Initial configuration

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

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

For more details, see *Configuring the PEM server on Linux* .

Note

- The operating system user pem is created while installing the PEM server. The PEM application data and the session is saved to this user's home directory.

Supported locales

Currently, the Postgres Enterprise Manager server and web interface support a locale of `English(US) en_US` and use of a period (.) as a language separator character. Using an alternate locale or a separator character other than a period might cause errors.

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

Operating system-specific install instructions are described in the corresponding documentation:

Red Hat Enterprise Linux (RHEL) and derivatives

- *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 and derivatives

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

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

You can install PEM on a single server, or you can install the web application server and the backend database on two separate servers. You must prepare your servers for PEM installation.

After fulfilling the prerequisites and completing the installation procedure described in the following steps, you must *Configure the web server (NGINX)* PEM. If you're using two servers, install and configure PEM on both servers.

Prerequisites

Before you begin the installation process:

1. Install a supported Postgres instance for PEM to use as a backend database.

You can install this instance on the same server to be used for the PEM web application or on a separate server. You can also use an existing Postgres instance if it is configured as detailed in the next steps.

2. Configure authentication on the Postgres backend database by updating the `pg_hba.conf` file.

Make the following changes manually, prior to configuration. (Additional changes are necessary during *Modify the SELinux configuration* .)

- To create the relations required for PEM, the PEM configuration script connects to the Postgres backend database as a superuser of your choice using password authentication. This requires you to permit your chosen superuser to authenticate using a password. This user must be able to connect from any location where you run the configuration script. In practice, this means the server where the backend database is located and the server where the PEM web application is to be installed, if they're different.
- To allow the chosen superuser to connect using password authentication, add a line to `pg_hba.conf` that allows `host` connections using `md5` or `scram-sha-256` authentication, such as `host all superusername 127.0.0.1/32 scram-sha-256` .

> **Note**
>
> If you're using EDB Postgres Advanced Server, see Modifying the pg_hba.conf file .

If you're using PostgreSQL, see Client Authentication . .. ::
2_blank_1_3. Verify that the *sslutils* extension is installed on your Postgres server.

If you're using PostgreSQL or EDB Postgres Extended Server on RHEL/AlmaLinux/Rocky Linux or SLES, you also need to install the `hstore contrib` module.

- If you're using EDB Postgres Advanced Server, you can install the `sslutils` extension as follows, where `<x>` is the EDB Postgres Advanced server version.

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

- If you're using PostgreSQL, you can install the `sslutils` and, if required, `hstore` modules as follows, where `<x>` is the PostgreSQL version.

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

- If you're using EDB Postgres Extended Server, you can install the `sslutils` and, if required, `hstore` modules as follows, where `<x>` is the EDB Postgres Extended Server version.

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

4. If you're using a firewall, allow access to port 8443 on the server where the PEM web application will be located:

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

firewall-cmd --reload
```

5. Make sure the components Postgres Enterprise Manager depends on are up to date on all servers. You can do this by updating the whole system using your package manager as shown below. If you prefer to update individual packages, a full list of dependencies is provided in *Dependencies of the PEM Server and Agent on Linux* .

```
sudo dnf upgrade
```

Install the package

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

Initial configuration

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

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

For more details, see *Configuring the PEM server on Linux* .

Note

- The operating system user pem is created while installing the PEM server. The PEM application data and the session is saved to this user's home directory.

Supported locales

Currently, the Postgres Enterprise Manager server and web interface support a locale of `English(US) en_US` and use of a period (.) as a language separator character. Using an alternate locale or a separator character other than a period might cause errors.

Installing Postgres Enterprise Manager server on Debian 12 arm64

You can install PEM on a single server, or you can install the web application server and the backend database on two separate servers. You must prepare your servers for PEM installation.

After fulfilling the prerequisites and completing the installation procedure described in the following steps, you must *Configure the web server (NGINX)* PEM. If you're using two servers, install and configure PEM on both servers.

Prerequisites

Before you begin the installation process:

1. Install a supported Postgres instance for PEM to use as a backend database.

You can install this instance on the same server to be used for the PEM web application or on a separate server. You can also use an existing Postgres instance if it is configured as detailed in the next steps.

2. Configure authentication on the Postgres backend database by updating the `pg_hba.conf` file.

Make the following changes manually, prior to configuration. (Additional changes are necessary during *Modify the SELinux configuration* .)

- To create the relations required for PEM, the PEM configuration script connects to the Postgres backend database as a superuser of your choice using password authentication. This requires you to permit your chosen superuser to authenticate using a password. This user must be able to connect from any location where you run the configuration script. In practice, this means the server where the backend database is located and the server where the PEM web application is to be installed, if they're different.

- To allow the chosen superuser to connect using password authentication, add a line to `pg_hba.conf` that allows `host` connections using `md5` or `scram-sha-256` authentication, such as `host all superusername 127.0.0.1/32 scram-sha-256`.

> **Note**
>
> If you're using EDB Postgres Advanced Server, see Modifying the pg_hba.conf file .

If you're using PostgreSQL, see Client Authentication . .. ::

2_blank_1_3. Verify that the *sslutils* extension is installed on your Postgres server.

If you're using PostgreSQL or EDB Postgres Extended Server on RHEL/AlmaLinux/Rocky Linux or SLES, you also need to install the `hstore contrib` module.

- If you're using EDB Postgres Advanced Server, you can install the `sslutils` extension as follows, where `<x>` is the EDB Postgres Advanced server version.

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

- If you're using PostgreSQL, you can install the `sslutils` and, if required, `hstore` modules as follows, where `<x>` is the PostgreSQL version.

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

- If you're using EDB Postgres Extended Server, you can install the `sslutils` and, if required, `hstore` modules as follows, where `<x>` is the EDB Postgres Extended Server version.

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

4. If you're using a firewall, allow access to port 8443 on the server where the PEM web application will be located:
5. Make sure the components Postgres Enterprise Manager depends on are up to date on all servers. You can do this by updating the whole system using your package manager as shown below. If you prefer to update individual packages, a full list of dependencies is provided in *Dependencies of the PEM Server and Agent on Linux* .

```
sudo apt-get upgrade
```

Install the package

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

Initial configuration

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

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

For more details, see *Configuring the PEM server on Linux* .

> **Note**
>
> - The operating system user pem is created while installing the PEM server. The PEM application data and the session is saved to this user's home directory.

Supported locales

Currently, the Postgres Enterprise Manager server and web interface support a locale of `English(US) en_US` and use of a period (.) as a language separator character. Using an alternate locale or a separator character other than a period might cause errors.

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

Operating system-specific install instructions are described in the corresponding documentation:

Red Hat Enterprise Linux (RHEL)

- *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

You can install PEM on a single server, or you can install the web application server and the backend database on two separate servers. You must prepare your servers for PEM installation.

After fulfilling the prerequisites and completing the installation procedure described in the following steps, you must *Configure the web server (NGINX)* PEM. If you're using two servers, install and configure PEM on both servers.

Prerequisites

Before you begin the installation process:

1. Install a supported Postgres instance for PEM to use as a backend database.

You can install this instance on the same server to be used for the PEM web application or on a separate server. You can also use an existing Postgres instance if it is configured as detailed in the next steps.

2. Configure authentication on the Postgres backend database by updating the `pg_hba.conf` file.

Make the following changes manually, prior to configuration. (Additional changes are necessary during *Modify the SELinux configuration* .)

- To create the relations required for PEM, the PEM configuration script connects to the Postgres backend database as a superuser of your choice using password authentication. This requires you to permit your chosen superuser to authenticate using a password. This user must be able to connect from any location where you run the configuration script. In practice, this means the server where the backend database is located and the server where the PEM web application is to be installed, if they're different.
- To allow the chosen superuser to connect using password authentication, add a line to `pg_hba.conf` that allows `host` connections using `md5` or `scram-sha-256` authentication, such as `host all superusername 127.0.0.1/32 scram-sha-256` .

> **Note**
>
> If you're using EDB Postgres Advanced Server, see Modifying the pg_hba.conf file .

If you're using PostgreSQL, see Client Authentication . .. ::
2_blank_1_3. Verify that the *sslutils* extension is installed on your Postgres server.

If you're using PostgreSQL or EDB Postgres Extended Server on RHEL/AlmaLinux/Rocky Linux or SLES, you also need to install the `hstore contrib` module.

- If you're using EDB Postgres Advanced Server, you can install the `sslutils` extension as follows, where `<x>` is the EDB Postgres Advanced server version.

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

- If you're using PostgreSQL, you can install the `sslutils` and, if required, `hstore` modules as follows, where `<x>` is the PostgreSQL version.

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

- If you're using EDB Postgres Extended Server, you can install the `sslutils` and, if required, `hstore` modules as follows, where `<x>` is the EDB Postgres Extended Server version.

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

4. If you're using a firewall, allow access to port 8443 on the server where the PEM web application will be located:

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

firewall-cmd --reload
```

5. Make sure the components Postgres Enterprise Manager depends on are up to date on all servers. You can do this by updating the whole system using your package manager as shown below. If you prefer to update individual packages, a full list of dependencies is provided in *Dependencies of the PEM Server and Agent on Linux* .

```
sudo dnf upgrade
```

Install the package

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

Initial configuration

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

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

For more details, see *Configuring the PEM server on Linux* .

Note

- The operating system user pem is created while installing the PEM server. The PEM application data and the session is saved to this user's home directory.

Supported locales

Currently, the Postgres Enterprise Manager server and web interface support a locale of `English(US) en_US` and use of a period (.) as a language separator character. Using an alternate locale or a separator character other than a period might cause errors.

Installing Postgres Enterprise Manager server on RHEL 8 ppc64le

You can install PEM on a single server, or you can install the web application server and the backend database on two separate servers. You must prepare your servers for PEM installation.

After fulfilling the prerequisites and completing the installation procedure described in the following steps, you must *Configure the web server (NGINX)* PEM. If you're using two servers, install and configure PEM on both servers.

Prerequisites

Before you begin the installation process:

1. Install a supported Postgres instance for PEM to use as a backend database.

You can install this instance on the same server to be used for the PEM web application or on a separate server. You can also use an existing Postgres instance if it is configured as detailed in the next steps.

2. Configure authentication on the Postgres backend database by updating the `pg_hba.conf` file.

Make the following changes manually, prior to configuration. (Additional changes are necessary during *Modify the SELinux configuration* .)

- To create the relations required for PEM, the PEM configuration script connects to the Postgres backend database as a superuser of your choice using password authentication. This requires you to permit your chosen superuser to authenticate using a password. This user must be able to connect from any location where you run the configuration script. In practice, this means the server where the backend database is located and the server where the PEM web application is to be installed, if they're different.
- To allow the chosen superuser to connect using password authentication, add a line to `pg_hba.conf` that allows `host` connections using `md5` or `scram-sha-256` authentication, such as `host all superusername 127.0.0.1/32 scram-sha-256` .

> **Note**
>
> If you're using EDB Postgres Advanced Server, see Modifying the pg_hba.conf file .

If you're using PostgreSQL, see Client Authentication . .. ::
2_blank_1_3. Verify that the *sslutils* extension is installed on your Postgres server.

If you're using PostgreSQL or EDB Postgres Extended Server on RHEL/AlmaLinux/Rocky Linux or SLES, you also need to install the `hstore contrib` module.

- If you're using EDB Postgres Advanced Server, you can install the `sslutils` extension as follows, where `<x>` is the EDB Postgres Advanced server version.

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

- If you're using PostgreSQL, you can install the `sslutils` and, if required, `hstore` modules as follows, where `<x>` is the PostgreSQL version.

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

- If you're using EDB Postgres Extended Server, you can install the `sslutils` and, if required, `hstore` modules as follows, where `<x>` is the EDB Postgres Extended Server version.

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

4. If you're using a firewall, allow access to port 8443 on the server where the PEM web application will be located:

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

firewall-cmd --reload
```

5. Make sure the components Postgres Enterprise Manager depends on are up to date on all servers. You can do this by updating the whole system using your package manager as shown below. If you prefer to update individual packages, a full list of dependencies is provided in *Dependencies of the PEM Server and Agent on Linux* .

```
sudo dnf upgrade
```

Install the package

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

Initial configuration

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

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

For more details, see *Configuring the PEM server on Linux* .

Note

- The operating system user pem is created while installing the PEM server. The PEM application data and the session is saved to this user's home directory.

Supported locales

Currently, the Postgres Enterprise Manager server and web interface support a locale of `English(US) en_US` and use of a period (.) as a language separator character. Using an alternate locale or a separator character other than a period might cause errors.

Installing Postgres Enterprise Manager server on SLES 15 ppc64le

Note

Postgres Enterprise Manager 8.3 and later is supported on SLES.

Prerequisites

Before you begin the installation process:

1. Install a supported Postgres instance for PEM to use as a backend database.

You can install this instance on the same server to be used for the PEM web application or on a separate server. You can also use an existing Postgres instance if it is configured as detailed in the next steps.

2. Configure authentication on the Postgres backend database by updating the `pg_hba.conf` file.

Make the following changes manually, prior to configuration. (Additional changes are necessary during *Modify the SELinux configuration* .)

- To create the relations required for PEM, the PEM configuration script connects to the Postgres backend database as a superuser of your choice using password authentication. This requires you to permit your chosen superuser to authenticate using a password. This user must be able to connect from any location where you run the configuration script. In practice, this means the server where the backend database is located and the server where the PEM web application is to be installed, if they're different.
- To allow the chosen superuser to connect using password authentication, add a line to `pg_hba.conf` that allows `host` connections using `md5` or `scram-sha-256` authentication, such as `host all superusername 127.0.0.1/32 scram-sha-256`.

> **Note**
>
> If you're using EDB Postgres Advanced Server, see Modifying the pg_hba.conf file .

If you're using PostgreSQL, see Client Authentication . .. ::

2_blank_1_3. Verify that the *sslutils* extension is installed on your Postgres server.

If you're using PostgreSQL or EDB Postgres Extended Server on RHEL/AlmaLinux/Rocky Linux or SLES, you also need to install the `hstore contrib` module.

- If you're using EDB Postgres Advanced Server, you can install the `sslutils` extension as follows, where `<x>` is the EDB Postgres Advanced server version.

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

- If you're using PostgreSQL, you can install the `sslutils` and, if required, `hstore` modules as follows, where `<x>` is the PostgreSQL version.

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

- If you're using EDB Postgres Extended Server, you can install the `sslutils` and, if required, `hstore` modules as follows, where `<x>` is the EDB Postgres Extended Server version.

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

4. If you're using a firewall, allow access to port 8443 on the server where the PEM web application will be located:

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

firewall-cmd --reload
```

5. Make sure the components Postgres Enterprise Manager depends on are up to date on all servers. You can do this by updating the whole system using your package manager as shown below. If you prefer to update individual packages, a full list of dependencies is provided in *Dependencies of the PEM Server and Agent on Linux* .

```
sudo zypper update
```

Install the package

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

Initial configuration

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

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

For more details, see *Configuring the PEM server on Linux* .

Note

- The operating system user pem is created while installing the PEM server. The PEM application data and the session is saved to this user's home directory.

Supported locales

Currently, the Postgres Enterprise Manager server and web interface support a locale of `English(US) en_US` and use of a period (.) as a language separator character. Using an alternate locale or a separator character other than a period might cause errors.

7.4.5 Installing Postgres Enterprise Manager server on Windows

On Windows, the PEM server graphical installer installs and configures the PEM server, a PEM agent, and the software required to connect to the PEM web interface with your choice of browser.

At the heart of each PEM installation is the PEM server. In a production environment, the server typically is a dedicated machine, monitoring a large number of Postgres servers or a smaller number of busy servers.

The PEM server backend database can be an EDB distribution of the PostgreSQL or EDB Postgres Advanced Server database server or an existing Postgres server installed from another source. The Postgres backend database server must be version 13 or later and contains a database named `pem` , which is used by the PEM server as a repository.

- If you want to use an existing Postgres server to host the PEM server, the PEM server installer can create the `pem` database on the Postgres host. You must manually satisfy the software prerequisites if you choose to use an existing server.

For more information about using an existing Postgres server to host the PEM server backend database, see *Installing the PEM server on an existing Postgres servers* .

- If you don't want to use an existing installation of Postgres as the PEM server host, the PEM server installer can:
- Install PostgreSQL
- Satisfy the server host's software prerequisites
- Create an instance (a PostgreSQL database cluster) that contains the `pem` database, which is the simplest PEM server installation option
- PEM-HTTPD is made available for Postgres installations through the PEM server installer or the StackBuilder utility. If PEM-HTTPD is already installed on the host, the PEM server installer reviews and updates the existing installation if needed. If the PEM server host doesn't contain an existing PEM-HTTPD installation, the PEM server installer adds it.
- Before installing the PEM server, you must decide if you want to run PostgreSQL and PEM-HTTPD on the same host or on separate hosts. If you intend to run the PostgreSQL database server and PEM-HTTPD on different hosts, then you must run the PEM server installer twice, once on each host. See Installing the PEM server and PEM-HTTPD on separate hosts .

- For detailed information about installing the PEM server and PEM-HTTPD on the same host, see Installing the PEM server and PEM-HTTPD on the same host .
- For detailed information about installing and configuring a standalone PEM agent, see *Installing Postgres Enterprise Manager agent on Windows* .
- Language pack installers contain supported languages that you can use with EDB Postgres Advanced Server and EDB PostgreSQL database installers. The language pack installer allows you to install Perl, TCL/TK, and Python without installing supporting software from third-party vendors. For more information about installing and using the language pack, see EDB Postgres language pack .
- For troubleshooting the installation or configuration of the PEM agent, see *Troubleshooting agent issues* .

The PEM server installer also installs the software required to access the server using the PEM web interface. You can access the web interface with a supported version of your browser. You can use the web interface to:

- Review information about objects that reside on monitored servers
- Manage databases and database objects that reside on monitored servers
- Review statistical information gathered by the PEM server

Note

If you're using SSL certificates, make sure that all the SSL certificates are in the data directory in the backend database server. If the certificates aren't in the data directory, then the PEM server's configuration might fail because it looks in the data directory while configuring the PEM server.

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

The easiest PEM server installation configuration consists of a PEM backend database server (hosted on a PostgreSQL database installed with the PEM server installer) and a PEM-HTTPD service that reside on the same host. In this configuration, the PEM server installer provides the prerequisite software for the backend host to register the service (on Windows).

1. Invoke the PEM server installer. On a Windows system, right-click the installer icon and select **Run as Administrator**. The installer displays a Welcome screen. Select **Next**.
2. Review the license agreement before selecting the appropriate radio button and accepting the agreement. Select **Next**.
3. Use the Installation Directory dialog box to specify the location of the PEM server:

- By default, the PEM server is installed in `C:\Program Files\edb\pem` on Windows. Accept the default location, or use the **Installation Directory** button to open a dialog box and select the directory to install the PEM server in.
- Use the **Show advanced options** check box to open the Advanced Options dialog box during installation. Use the Advanced Options dialog box when installing the Postgres database server and the PEM-HTTPD on different hosts or if you want the PEM server to reside on an existing Postgres server installation.
- To install the PostgreSQL server package with the installer and PEM-HTTPD on the same host, clear **Show advanced options** and select **Next**.

The PEM server installer performs a preinstallation check for PEM-HTTPD, Language Pack, and PostgreSQL 13. If the installer doesn’t locate these packages, it informs you in the Dependency Missing dialog box.

> **Note**
>
> By default EDB Language Pack is installed in *C:edblanguagepackv1* .

1. If the dependencies are missing, the PEM server installer launches the respective installation wizards. Follow the wizard's onscreen directions each package.

After installing any missing dependencies, the installation process continues by displaying the Database Server Installation Details dialog box. The information provided on the Database Server Installation Details dialog box enables the installer to connect to the PostgreSQL server.

1. Provide the user name and password of a database superuser. After supplying the requested information, select **Next**.
2. After providing the name and password of the Postgres database superuser, you might be prompted for the password to the user account under which the PEM agent will run. If prompted, provide the password, and select **Next**.
3. Use the Network Details dialog box to specify the CIDR-style network address from which the PEM agents will connect to the server (the *client-side* address).

You can specify the address of a network host or a network address range. For example, if you want to monitor database servers with the addresses `192.168.10.23` , `192.168.10.76` , and `192.168.10.184` , enter `192.168.10.0/24` to allow connections with hosts in that network.

The specified address is added to the server's `pg_hba.conf` file. You can specify more network addresses by manually adding entries to the `pg_hba.conf` file on the PostgreSQL server. Use the first entry as a template.

After you add the network address, select **Next**.

The PEM server installer installs a PEM agent on the host where the server resides to monitor the server and provide alert processing and garbage collection services. A certificate is also installed in the location specified in the **Agent certificate path** field.

1. Enter an alternate description or select an alternate agent certificate path for the PEM agent, or accept the defaults. Select **Next**.
2. You must enter your organization details to be used in SSL certificates generated by PEM.
3. The wizard is now ready to install the PEM server. Select **Next** to continue with the installation.

During the installation process, the installer copies files to the system and sets up the database and web services required to run PEM. When the installation completes, a confirmation indicates that: - The web service was configured. - The web service is listening on port 8443. - The `pem` database was created and configured.

1. Select **OK**.

Installing the PEM server and PEM-HTTPD on separate hosts

To use separate hosts for the PEM server backend database and PEM-HTTPD:

1. Invoke the PEM server installer on the host of the Postgres server that will contain the `pem` database. During the installation, select the **Database** option on the Advanced Options dialog box, and provide connection information for the Postgres server.
2. Modify the `pg_hba.conf` file of the Postgres installation on which the PEM server (and `pem` database) resides, allowing connections from the host of the PEM-HTTPD server.
3. Invoke the PEM server installer on the host of the PEM-HTTPD server, selecting the **Web Services** option on the Installation Type dialog box.

To start the installation:

1. On a Windows system, invoke the PEM server installer. Right-click the installer and select **Run as Administrator**. The installer displays a Welcome screen. Select **Next**.
2. Review the license agreement before selecting the appropriate radio button and accepting the agreement. Select **Next**.
3. Use fields on the Installation Directory dialog box to specify the directory for the PEM server to reside and to open the Advanced Options dialog box during installation:

- By default, the PEM server is installed in `C:\Program Files\edb\pem` on Windows. Accept the default location, or use the **Installation Directory** field to open a browser dialog box and select the directory to install the PEM server in.
- To install the PEM server and PEM-HTTPD on separate hosts, use the Advanced Options dialog box to specify the installation type (**Web Services** or **Database**). Select **Show advanced options** to include the Advanced Options dialog box in the installation process.
- Select **Next**.

1. Use the Advanced Options dialog box to specify the components that you want to install:

- Select **Web Services and Database** to indicate that the Postgres server and PEM-HTTPD will both reside on the current host. If you select **Web Services and Database**, the PEM server installer lets you specify the Postgres server to use for the PEM server before checking for a PEM-HTTPD installation.
- Select **Web Services** to install PEM-HTTPD on the current host, while using a Postgres database server that resides on another host to host the PEM server and `pem` database.

Note

Before using the **Web Services** option to install PEM-HTTPD, you must complete the PEM server installation process on the host of the PEM server and *pem* backend database. Select **Database** on the Advanced Options dialog box and modify the connection properties of the *pg_hba.conf* file on the PEM server.

This option invokes the installation steps described in *Installing web services* .

- Select **Database** to use an existing Postgres server (version 13 or later) or to install only the database server that's distributed with the PEM server installer. This option invokes the installation steps described in *Specifying a database host* .

1. After selecting an installation option, select **Next**.

Specifying a database host

1. Select **Database** on the Advanced Options dialog box to specify connection information for the host where the PEM server backend database (named `pem`) will reside. Select **Next**.
2. Use the list on the Database Server Selection dialog box to select a host for the PEM server backend database. You can:

- Select a host from existing Postgres installations that reside on the current host.

Note

You might need to add the *sslutils* package to your installation.`2_blank_4_-` Select **PostgreSQL** to install the Postgres server that's distributed with the PEM server installer, where <*x*> is the PostgreSQL database server version. If you decide to use the version of PostgreSQL that's bundled with the PEM server installer, the EnterpriseDB one-click PostgreSQL installer opens and walks you through the installation.`1_blank_4_-` Select **Other**

Database Server to specify connection information for a Postgres server that wasn't installed using a one-click graphical installer from EnterpriseDB. *1_blank_4_*- Select **Next**.

If the PEM server will reside on an existing Postgres server, the Database Server Installation Details dialog box opens.

1. The information required on the Database Server Installation Details dialog box can vary. The PEM server installer asks you to provide only the information about the selected installation that it can't locate:

- Specify the name of a Postgres database superuser in the **User** field.
- Specify the password associated with that user in the **Password** field.
- Select **Next**.

1. If prompted, provide the system password for the service account under which the PEM agent will run. Select **Next**.
2. Use the Network Details dialog box to specify the CIDR-style network address from which PEM agents will connect to the server (the client-side address). The specified address is added to the server's `pg_hba.conf` file.

Note

You can specify more network addresses by manually adding entries to the *pg_hba.conf* file on the PostgreSQL server.

Accept the default (specifying the localhost), or specify a **Network address** range. Select **Next**.

The PEM server installer installs a PEM agent on the host on which the server resides to monitor the server and provide alert processing and garbage collection services. A certificate is also installed in the location specified in the **Agent certificate path** field.

1. You can enter an alternate description or an alternate agent certificate path for the PEM agent or accept the defaults. Select **Next**.
2. The wizard is now ready to install the PEM server. Select **Next** to proceed with the installation.

During the installation process, the installer copies files to the system and sets up the PEM server's backend database. A confirmation indicates that the `pem` database was created and configured.

1. Select **OK**

When the database portion of the PEM server installation is complete, you can invoke the PEM server on another host to install or upgrade PEM-HTTPD.

Installing web services

Select **Web Services** on the Advanced Options dialog box to either install PEM-HTTPD on the current host or update an existing PEM-HTTPD installation.

Note

The current host might not be the host of the PEM backing database.

Before selecting this option, you must:

- Install the PEM server installer on a host system, during which you specify a backing database for the PEM server.

- Modify the `pg_hba.conf` file on the PEM server database host to allow connections from the PEM-HTTPD host.
- Restart the database server.

1. Select **Web Services** and select **Next**. The PEM server installer checks the current host for existing PEM-HTTPD and LanguagePack installations.
2. If the installer doesn't locate the components, the installer informs you that one or more dependencies are missing. Select **Next** to install the language pack. After installing language pack, the installer invokes the PEM-HTTPD setup wizard.
3. Follow the onscreen instructions of the PEM-HTTPD Setup wizard. When the wizard completes the installation, select **Finish** to open the Database Server Installation Details dialog box.
4. Use the Database Server Installation Details dialog box to provide connection information for the Postgres installation that's hosting the PEM server installation:

- Enter the name or IP address of the PEM server host in the **Host** field.
- Enter the port number on which the Postgres server is listening in the **Port** field.
- Enter the name of a Postgres database superuser in the **User** field.
- Enter the password associated with the Postgres superuser in the **Password** field.

1. Select **Next**.

Before completing the PEM server installation, the installer contacts the database host. For the installation to continue, the `pg_hba.conf` file on the PEM database host must be configured to accept connections from the host of the httpd server, and the firewall must allow a connection. The PEM server installer completes the PEM server installation, adding only those items that must reside on the host of the PEM-HTTPD server.

Installing the PEM server on an existing Postgres servers

You can use an existing Postgres server (version 13 or later) to host the PEM server and the `pem` database. Postgres installers and prerequisite software extensions are freely available on the EnterpriseDB website .

You can configure an existing Postgres server for a PEM server installation.

> **Note**
>
> These instructions are guidelines. the actual steps needed to configure your Postgres installation depend on the configuration of your Postgres server.

Graphical installation

Invoke the PEM server installer. Assume administrator privileges and navigate to the directory that contains the installer. Then, invoke the installer:

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

Where `<x>` is the major and minor versions of PEM, and `<platform>` is the platform.

Then, begin the installation:

1. The installer displays a Welcome screen. Select **Next**.
2. Review the license agreement before selecting the appropriate radio button and accepting the agreement. Select **Next**.

3. Use the Installation Directory dialog box to specify the location of the PEM server and access the Advanced Options dialog box:

- Use the **Installation Directory** field to open a browser dialog and select the directory to install the PEM server in.
- If you're installing the PEM server on an existing server, select **Show advanced options** to include the Advanced Options dialog box in the installation process.
- Select **Next**.

1. Use the Advanced Options dialog box to specify an installation type. Select:

- **Web Services and Database** if both the Postgres server and the PEM-HTTPD server will reside on the current host. This option is valid if you're using an existing Postgres server to host the PEM server or using the PEM server installer to install the Postgres server where the PEM server will reside.

If you select **Web Services and Database**, the PEM server installer checks the current host for a PEM-HTTPD installation and upgrades or installs PEM-HTTPD if necessary.

- **Web Services** if only the PEM-HTTPD server will reside on the current host. See Installing web services for more information about invoking this option.
- **Database** if you're installing only the PEM server (and creating the `pem` backend database) on the current host. This option is valid if you're using an existing Postgres server to host the PEM server or using the PEM server installer to install the PostgreSQL server where PEM will reside.

After selecting an installation option, select **Next**.

1. Use the list on the Database Server Selection dialog box to select a backend database for the PEM server:

- Select the name of a Postgres server on the current host that was installed using a Postgres one-click installer or EDB Postgres Advanced Server installer.
- Select **PostgreSQL x (Packaged)** to install and use the PostgreSQL server that's packaged with the PEM server installer, where `x` is the version of the PostgreSQL database server.
- Select **Other Database Server** to to use a Postgres database that was installed from a source other than an EnterpriseDB installer (such as from an rpm or built from source).

> **Note**
>
> The selected database server must include an installation of the *sslutils* contrib module and a registered service (on Windows).

If you selected **Web Services and Database** on the Advanced Options dialog box, the installation wizard checks the current host for an existing PEM-HTTPD installation and upgrades or installs the service as needed.

If you selected **Database** on the Advanced Options dialog box, the Database Server Installation Details dialog box opens.

2. Use the fields on the Database Server Installation Details dialog box to describe the connection to the Postgres server that will host the PEM server:

- Enter the name of a database superuser in the **User** field.
- Enter the password associated with the superuser in the **Password** field.

Select **Next**.

3. Provide the administrator password for the PEM agent service to run under. Select **Next**.

4. Use the Network Details dialog box to specify the CIDR-style network address for the PEM agents to connect to the server (the `client-side` address). The specified address is added to the server's `pg_hba.conf file.`

You can specify more network addresses by manually adding entries to the `pg_hba.conf` file on the PostgreSQL server. Use the first entry as a template.

After you add the network address, select **Next**.

The PEM server installer installs a PEM agent to the host where the server resides to monitor the server and provide alert processing and garbage collection services. A certificate is also installed in the location specified in the **Agent certificate path** field.

5. You can enter an alternate description or an alternate agent certificate path for the PEM agent or accept the defaults. Select **Next**.
6. You must enter your organization details to be used in SSL certificates generated by PEM.
7. The wizard is now ready to install the PEM server. Select **Next** to continue with the installation.

During the installation process, the installer copies files to the system and sets up the PEM server's backend database. A confirmation indicates that the `pem` database was created and configured.

8. Select **OK**.

If you're using a PEM-HTTPD service that resides on a separate host, you must:

- Modify the `pg_hba.conf` file on the Postgres server host to allow connections between the hosts.
- Invoke the PEM server installer on the host of the PEM-HTTPD server. See Installing Web Services for more information about installing PEM-HTTPD.

Invoking the server installer from the command line

The command line options of the PEM server and PEM agent installers offer functionality in situations where a graphical installation might not work because of limited resources or system configuration. You can:

- Include the `--mode unattended` option when invoking the installer to perform an installation without user input.

Not all command line options are suitable for all platforms. For a complete reference guide to the command line options, use the `--help` option when you invoke the installer.

Invoking the PEM server installer in unattended mode

You can perform an unattended PEM server installation by providing installation preferences on the command line when invoking the installer. The system on which you're installing the PEM server must have internet access.

You must have administrator privileges to install the PEM server. Before invoking the PEM server installer, you must install the following dependencies:

- PostgreSQL
- pem-httpd
- Language Pack

You can use the PEM server installer to satisfy the dependencies of the PEM server. Navigate to the location of the installer, and use the following command to extract the dependencies:

```
pem-server-10.<x>.<x>-windows-x64.exe --extract-dependents C:\
```

In this example, the files are extracted to the `C:\` directory. After extracting the files, you must install each program. Navigate to the directory that contains the files (in this example, `C:\)` , and enter:

```
edb-languagepack-<version>-windows-x64.exe --mode unattended
pem-httpd-<version>-windows-x64.exe --mode unattended
postgresql-<version>-windows-x64.exe --mode unattended
```

Then, you can invoke the PEM server installer:

```
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
```

Where:

- `registered_edb_user` specifies the name of a registered EnterpriseDB user. To register, visit the EDB website .
- `edb_user_password` specifies the password associated with the EDB user account.
- `port` specifies the port used by the backing PostgreSQL database. By default, the PostgreSQL database uses port 5432.
- `cidr_address_range` specifies the address range to add to the `pg_hba.conf` file of the PEM server's backing database to allow connections from the agents for the server to monitor. You can specify a network range (for example, 192.168.2.0/24) to provide server access to agents that reside on the same network.
- `windows_password` specifies the password associated with the Windows administrator's account.

Note

When invoked in unattended mode, the PostgreSQL installer creates a user named *postgres* with a password of *postgres* .

7.4.6 Creating an EDB repository on an isolated network

You can create a local repository to act as a host for the PEM RPM packages if the server where you want to upgrade PEM can't directly access the EDB repository. This is a high-level overview of the steps required. You might need to modify the process for your own network.

To create and use a local repository:

1. Use the following commands on a system with Internet access to download the dependencies for PEM:

```
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*
```

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

2. Copy the directories /<pem_dir> and /<epel_dir> to the machine in the isolated network.
3. Create the repositories:

```
createrepo /<pem_dir>
createrepo /<epel_dir>
```

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

```
[pemrepo]
name=PEM Repository
baseurl=file:///pem7/
enabled=1
gpgcheck=0
```

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

```
[epelrepo]
name=epel Repository
baseurl=file:///pem7/
enabled=1
gpgcheck=0
```

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

To install PEM server:

```
dnf install edb-pem
```

To upgrade PEM server:

```
dnf upgrade edb-pem
```

7.4.7 Configuring the PEM server on Linux

Configuring the PEM Server

The PEM server package includes a script (configure-pem-server.sh) to help automate the configuration process for Linux platform installations. The script is installed in the /usr/edb/pem/bin directory. To invoke the script, run the following command as root:

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

Note

If you're providing SSL certificates, make sure that all the certificates are in the data directory of the backend database server. If the certificates aren't in the data directory, then the PEM server's configure script might fail because it looks into the data directory while configuring the PEM server.

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

Short Option	Long Option	Description
-acp	*--pemagent-certificate-path*	Defines PEM agent certificate path. The default is */root/.pem*.
-arp	*--agent-register-param*	Additional parameters to pass to *pemworker --register-agent*, for example *--enable-smtp=true* or *--display-name=xxx*. Multiple options must be separated by commas.
-aro	*--agent-register-overrider*	Override default agent configuration values, for example *agent_id=1* or *max_connections=3*. Multiple options must be separated by commas.
-au	*--use-agent-user*	PEM agent user name. Use this parameter to override the default behavior of generating individual, numbered agent users (e.g., agent1, agent2).
-bhn	*--asb-host-name*	Specifies the IP address or fully qualified domain name of the monitored server. If not specified, it takes the value of the *--host* parameter.
-ci	*--cidr-address*	CIDR-formatted network address range that agents connect to the server from, to be added to the server's *pg_hba.conf* file, for example, *192.168.1.0/24*. The default is *0.0.0.0/0*.
-dbi	*--db-install-path*	Directory for the database server installation, for example, */usr/edb/as17* for EDB Postgres Advanced Server or */usr/pgsql-17* for PostgreSQL.
-ds	*--db-unitfile*	Unit file name of the PEM database server. For EDB Postgres Advanced Server, the default file name is *edb-as-17*. For PostgreSQL, it's *postgresql-17*.
-h	*--help*	Lists all the available options while configuring the PEM server.
-ho	*--host*	Host address of the PEM database server. This can be a comma-separated list of IPs or hostnames if you are using an *Query information* . See the libpq documentation for more details on the syntax.
-kdc	*--skip-db-certificate*	Doesn't create or configure database SSL certificates. Use this when you have already configured SSL on the database server and want to provide your own certificates.
-kwc	*--skip-web-certificate*	Doesn't create web server certificates. Use this when you want to provide your own web server SSL certificates.
-nhc	*--no-hba-change*	Skips the changes to *pg_hba.conf* and *postgresql.conf* files.
-p	*--port*	Port number of the PEM database server. This can be a comma-separated list of ports if you are using an *Query information* . See the libpq documentation for more details on the syntax.
-ps	*--pemagent-servicename*	Service name of the pemagent. The default value is *pemagent*.

continues on next page

Table 5 – continued from previous page

Short Option	Long Option	Description
-scs	*--server-certificate-subject*	Provides the custom web server certificate subject. The format is */C=CountryCode/ST=State/L=City/O=Company/CN=Hostname/emailAddress=user@company.com*. Provide the *C=CountryCode* as *Alpha-2* code.
-sp	*--superpassword*	Superuser password of the PEM database server. This value is required.
-srp	*--server-register-param*	Additional parameters to pass to *pemworker --register-server*, for example *--efm-cluster-name=xxx* or *--config-dir=path*.
-su	*--superuser*	Superuser name of the PEM database server.
-t	*--type*	Installation type: Specify *1* if the configuration is for web services and backend database, *2* if you're configuring web services, or *3* if you're configuring the backend database. If you specify *3*, the database must reside on the local host.
-uac	*--use-agent-sslcert*	Path to an existing agent SSL certificate file to use instead of generating one.
-uak	*--use-agent-sslkey*	Path to an existing agent SSL key file to use instead of generating one.
-ues	*--use-existing-sslutils*	Use pre-existing sslutils to generate agent certificates. Only has an effect when used with *--skip-db-certificate* or *--use-trusted-ssl-db-certificates*.
-un	*--uninstall-pem-server*	Uninstalls the PEM server.
-utd	*--use-trusted-ssl-db-certificates*	An enterprise-level flag that enables *--skip-db-certificate* and *--no-hba-change* options.
	--replication-solution	Specifies the replication solution for HA deployments. Valid values are *patroni* or *efm*.
	--replication-role	Specifies the replication role of this host for HA deployments. Valid values are *primary* or *standby*.

Note

The *--asb-host-name* option does not apply to the following: - PostgreSQL socket (except when specifying a socket path on Linux, which must begin with /) - *PEM_MONITORED_SERVER_PASSWORD* (except as a required but unused value when a socket path is provided) - *--server-addr* (except as a default if *--asb-host-name* is not supplied)

If you don't provide configuration properties on the command line, the script prompts you for values. When you invoke the script, choose from:

1. `Web Services and Database` _ Select this option if the web server and database both reside on the same host as the PEM server.
2. `Web Services` _ Select this option if the web server resides on a different host from the PEM server.
3. `Database` _ Select this option to configure the PEM backend database for use by the PEM server. The specified database must reside on the local host.

> **Note**
>
> If the web server and the backend database (PEM server) reside on separate hosts, configure the database server first (option 3) and then web services (option 2). The script proceeds only if the backend database is configured before web services.

After selecting a configuration option, the script prompts you for configuration properties. When the script finishes, it creates the objects required by the PEM server or performs the configuration steps required. To view help for the script, use the command:

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

Selecting a web server

PEM supports both NGINX and Apache HTTPD as the server for the web application. New installations use NGINX by default. The web server is installed and configured by the `configure-pem-server.sh` script.

To select the web server, set the `USE_NGINX` environment variable before running the script:

- `USE_NGINX=1` — Use NGINX (default)
- `USE_NGINX=0` — Use Apache HTTPD

For example, to use Apache HTTPD:

```
USE_NGINX=0 /usr/edb/pem/bin/configure-pem-server.sh
```

You may change the web server after configuration by running `/usr/edb/pem/bin/switch-web-server.sh` .

Configuring SELinux

If your system has SELinux enabled (the default for RHEL-like systems) you must run the `configure-selinux.sh` on the server where the PEM web application is installed. You must specify the web selected web server, even if you are using the default. `USE_NGINX=1` for NGINX and `USE_NGINX=0` for Apache HTTPD.

```
USE_NGINX=1 /usr/edb/pem/bin/configure-selinux.sh
```

Post-configuration steps when web server and PEM backend database are installed separately

If you choose to run the web application server on a separate host to the backend database, you need to perform some additional manual steps before PEM is fully operational.

Make sure that the backend Postgres database accepts the connections from any user permitted to log in to PEM from the web application server. To achieve this, add this entry to `pg_hba.conf` :

```
host pem +pem_user <web_app_ip>/32 md5
```

Where `<web_app_ip>` is the IP address of the web application server.

Additionally, if the IP address of the web application server isn't within the network address range specified when the script is executed, you must add two entries to allow the PEM agent on this server to connect:

```
host pem +pem_agent <web_app_ip>/32 md5
host pem +pem_agent <web_app_ip>/32 cert
```

Where `<web_app_ip>` is the IP address of the web application server.

Accessing the PEM application

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

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

By default, the web services listen on port 8443. To change the port, see *Changing the default port* .

7.4.8 Installing Postgres Enterprise Manager in HA Patterns

Before deploying High Availability PEM, please familiarize yourself with the high-level guidance in High Availability Patterns for PEM Deployment .

For PEM 10.4 and later, the recommended process for deploying HA PEM is as follows:

- Deploy an HA Postgres cluster with a supported cluster manager.
- Prepare the cluster for PEM installation by adding the required SSL files and HBA rules.
- Install and configure PEM on the primary, then on the standbys.
- If using separate web servers, deploy PEM on the web servers.

In versions of PEM prior to 10.4, the process was more complex due to limitations in the `configure-pem-server.sh` script. It is therefore highly recommended that new deployments of HA PEM are performed with the latest version. However, the changes to `configure-pem-server.sh` are backward-compatible so previous processes will still run without adjustment.

The following pages provide detailed instructions for some example HA PEM architectures following the steps above. They are not intended to be an exhaustive guide to all possible patterns.

- *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*

In addition, the following guides to specific elements of HA PEM deployment are provided:

- *Storing user settings in the backend, accessible to all web application instances*

HA PEM using the C1 architecture with EFM and a Virtual IP

This page provides detailed instructions to install and configure a High Availability (HA) PEM deployment according to reference architecture C1.

This example uses EDB Failover Manager (EFM 5.2) for cluster management, EDB Postgres Advanced Server (EPAS 17) as the PEM backend database and Virtual IP (VIP) as the mechanism for routing traffic to the primary on RHEL like systems. Please see *High availability patterns for PEM deployment* to understand other options.

Also refer to *Deciding on an SSL model* to determine how you will provide SSL certificates and keys prior to starting installation.

- 172.16.161.200 - PEM Primary
- 172.16.161.201 - PEM Standby 1
- 172.16.161.202 - PEM Standby 2
- 172.16.161.203 - EFM Witness Node
- 172.16.161.245 - PEM VIP (used by agents and users to connect)

The following must use the VIP address:

- The PEM agent binding of the monitored database servers
- Accessing the PEM web client
- Accessing the webserver services

Deploying an EFM cluster

Before installing PEM, you should have a working EFM cluster on the hosts. This section provides instructions to deploy such a cluster. You do not have to follow these steps if you have your own processes to provision EFM clusters. However, if you plan to use sslutils you should ensure it is installed on your cluster before proceeding.

Initial package installation and Postgres configuration

Perform the following steps on all hosts. Postgres is not required for witnesses.

1. Install the following packages:

- EDB Postgres Advanced Server (backend database for PEM Server)
- 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. Initialize a Postgres database and start the service.

```
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. Open the following ports on the firewall:

- `5444` for EPAS
- `7800` for EFM

For example:

```
firewall-cmd --zone=public --add-port=5444/tcp --permanent
firewall-cmd --zone=public --add-port=7800/tcp --permanent
firewall-cmd --reload
```

Optional: install sslutils

If you wish PEM to act as a certificate authority and automatically issue client certificates, you will need to configure *Optional: install sslutils* . If you plan to provide your own certificates, then you may skip this step.

1. Install the sslutils package on all non-witness hosts.

```
dnf install edb-as17-server-sslutils
```

Set up the primary node for streaming replication

Perform these steps on the primary only.

1. Create the replication role:

```
CREATE ROLE repl REPLICATION LOGIN PASSWORD password;
```

Give the password of your choice.

2. Configure the following in the `postgresql.conf` file:

```
wal_level = replica
max_wal_senders = 10
wal_keep_size = 500
max_replication_slots = 10
```

For more information on configuring parameters for streaming replication, see the PostgreSQL documentation .

3. Add the following entries in the host-based authentication (`/var/lib/edb/as17/data/pg_hba.conf`) file to allow the replication user to connect from all the standbys:

```
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. Restart the EPAS server.

```
systemctl restart edb-as-17.service
```

Add a .pgpass file

Perform these steps on all non-witness nodes.

1. Create the `.pgpass` file in the home directory of the enterprisedb user and add the following content. Replace `<password>` with the password of the replication user created previously.

```
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. Set the permissions on the file to restrict access:

```
chmod 600 ~/.pgpass
```

Set up the standby nodes for streaming replication

Use the pg_basebackup utility to create replicas of the PEM backend database server on the standby servers.

1. Stop the service for EPAS on all the standby nodes:

```
systemctl stop edb-as-17.service
```

1. Remove the data directory of the database server on all the standby nodes:

```
su - enterprisedb

rm -rf /var/lib/edb/as17/data/*
```

1. Take a backup of the primary node on each of the standby nodes using pg_basebackup:

```
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. Start the EPAS database server on each of the standby nodes:

```
systemctl start edb-as-17
```

Set up EFM to manage failover

Perform the following steps to set up EFM:

1. On the primary, create a database user `efm` to connect to the database servers. Grant execute privileges on the functions related to WAL logs, and monitoring privileges, to the user. As a superuser:

```
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;
```

This change will be replicated to the standbys.

1. On the primary, add entries in `pg_hba.conf` to allow the `efm` database user to connect to the database server from all nodes. These entries will be replicated to the standbys.

```
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. Reload the configurations on all the database servers.

```
SELECT pg_reload_conf();
```

4. On all nodes, create an `efm.nodes` file using the sample file (`/etc/edb/efm-5.2/efm.nodes.in`), and give read-write access to the EFM OS user:

```
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. On the standby nodes, add the IP address and EFM port of the primary node in the `/etc/edb/efm-5.2/efm.nodes` file:

6. On all nodes, create the `efm.properties` file using the sample file (`/etc/edb/efm-5.2/efm.properties.in`). Grant read access to all users:

```
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. On any node, encrypt the `efm` database user's password (as defined in Step 1 above) using the `efm` utility, make a note of the output for the next step.

```
export EFMPASS=password
/usr/edb/efm-5.2/bin/efm encrypt efm --from-env
```

8. On all nodes, edit the following parameters in the `efm.properties` file. Replace `<encrypted-password>` with the output of the previous step. Replace `<ip-addr>` with the IP address of each node.

> **Note**
>
> If your hosts are not connected to the internet, replace the value of *ping.server* with the address of a reliable server accessible on your network that will respond to pings.

For more detail on EFM configuration please refer to the documentation . .. ::
edb_notranlate_20

9. On the witness node, set the value of the `is.witness` configuration parameter to `true` :

```
is.witness=true
```

10. On the primary node, enable and start the EFM service:

```
systemctl enable edb-efm-5.2
systemctl start edb-efm-5.2
```

11. On the primary node, allow the standbys to join the cluster:

```
/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. Enable and start the EFM service on the standby nodes and the EFM witness node:

```
systemctl enable edb-efm-5.2
systemctl start edb-efm-5.2
```

13. Check the EFM cluster status by running the following command on any node.

```
/usr/edb/efm-5.2/bin/efm cluster-status efm
```

The output should look like this:

```
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.
```

This status confirms that EFM is set up successfully and managing the failover for the PEM server.

In case of failover, any of the standbys are promoted as the primary node, and PEM agents connect to the new primary node. You can replace the failed primary node with a new standby using the procedure above.

Prepare the EFM cluster for PEM installation

Before installing PEM in HA mode, we must ensure that the cluster meets two requirements:

- SSL is configured so that the `cert` auth method can be used for connections from PEM agents to the PEM backend database.
- HBA rules are in place to allow agents and web application instances to connect to the PEM backend.

Prepare SSL files

Follow the instructions in *Generating SSL files for HA PEM backend hosts* .

Modify the EFM cluster to add SSL and HBA rules

Perform the following actions on each PEM node (excluding the witness).

1. Modify `postgresql.conf` to enable SSL and set the paths to the SSL files created above.

```
ssl = on
ssl_ca_file = root.crt
ssl_cert_file = server.crt
ssl_key_file = server.key
```

If you have a CRL, also set:

```
ssl_crl_file = root.crl
```

1. Modify `pg_hba.conf` to add the rules required by PEM. Add this block at the top of the file, above any existing content, adjusting the IP addresses to match your cluster.

```
# 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
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. If using a common agent user, add one additional line to allow the first agent to connect during PEM installation. This line can be removed later if required.

```
hostssl pem      common_agent_user  127.0.0.1/32           cert
```

1. Modify the HBA rules added in the previous sections (for replication and EFM) to `hostssl` instead of `host` to enforce SSL for all connections.

Restart the cluster

> **Note**
>
> By default, EFM will trigger a failover if the database service stops. During this rolling restart, stop the EFM agent before restarting Postgres to prevent an unintended failover. !!!

Perform a rolling restart of the cluster. On each non-witness host, starting with the standbys and concluding with the primary, do the following:

1. Stop the EFM agent service

```
systemctl stop edb-efm-5.2
```

1. Restart Postgres

```
systemctl restart edb-as-17
```

1. Start the EFM agent service

```
systemctl start edb-efm-5.2
```

Prepare agent users and certificates

If you are providing your own SSL certificates, follow these steps to prepare for PEM deployment. If you are using sslutils, you may skip this section.

On the primary, create the common agent user. Grant connect permission on `template1` to the common agent user. This is only required temporarily and can be revoked later.

```
CREATE USER common_agent_user;
GRANT CONNECT ON DATABASE template1 TO common_agent_user;
```

Generate a certificate and key pair for this user and save them on all the hosts.

PEM will automatically generate unique, numbered Postgres users (agent1 through agent3) for the agents deployed across your three hosts.

To support this setup, you must manually generate three distinct certificate/key pairs—one for each specific agent user—and install one unique pair on each corresponding host.

Install PEM on the primary

Perform the following steps on the primary host only.

1. Install the PEM package.

```
dnf install edb-pem --allowerasing
```

1. Create a superuser that can login using a password. This is the user we will use to install PEM. It will also become the agent-server binding user for the agents running on PEM hosts.

```
su - enterprisedb -c "psql -c \"create role pemsuper login superuser password your-
↪password-here;\""
```

1. To configure the PEM server, run the configuration script using the Database option (Option 3) and set the host to your Virtual IP (VIP).

To ensure the system is correctly set up for High Availability (HA), you must include the following flags:

- **``--replication-solution efm`` & ``--replication-role primary``** : These flags force the script into HA mode, ensuring compatibility with your failover manager.
- **``--skip-db-certificate``** : This prevents the script from attempting to automatically configure SSL, as you are managing certificates manually in an HA environment.

When using sslutils you must include `--use-existing-sslutils` . This ensures that while the script skips the database SSL setup, it still utilizes the sslutils extension to handle agent certificate generation.

```
/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 \
                                         --replication-role primary \
                                         --skip-db-certificate \
                                         --use-existing-sslutils \
                                         --db-install-path /usr/edb/as17/ \
                                         --db-unitfile edb-as-17
```

When using a common agent user you must supply the username and path to the certificate and key generated for that user.

```
/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 \
                                         --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
```

After running this command, grant the common agent user the pem_agent role and revoke its permissions on `template1`.

```
GRANT pem_agent to common_agent_user;
REVOKE CONNECT ON DATABASE template1 FROM common_agent_user;
```

You may also remove the HBA rule we added for the common agent user at this stage if desired as it will now match the `+pem_agent` rules.

When using your own certificate you should specify the agent_id as `1` and path to the certificate and key generated for the corresponding `agent1` user.

```
/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 \
                                         --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
```

You will be prompted for various additional details. For configuration options see, Configuring the PEM server on Linux .

1. Optionally, to synchronize PEM web application user preferences between instances, *configure central storage of user preferences* . At this stage, you can only complete the backend configuration. We will configure the web application later.

Install PEM on the standbys

Perform the following steps on the standby hosts only.

1. Install the PEM package.

```
dnf install edb-pem --allowerasing
```

1. Run the PEM configuration script, specifying option 3 (Database) and the VIP as the host. You must specify the `--replication-solution efm` and `--replication-role standby` options to ensure HA mode is used.

```
/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 \
                                         --replication-role standby \
                                         --db-unitfile edb-as-17 \
                                         --db-install-path /usr/edb/as17/
```

When using a common agent user you must supply the username and path to the certificate and key generated for that user.

```
/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 \
                                         --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
```

When using your own certificate you should specify the agent_id as the next integer `N` and path to the certificate and key generated for the corresponding `agentN` user.

```
/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 \
                                         --replication-role standby \
                                         --db-unitfile edb-as-17 \
```

(continues on next page)

(continued from previous page)

```
                                                --db-install-path /usr/edb/as17/ \
                                                --use-agent-sslcert /path/to/agent/cert \
                                                --use-agent-sslkey /path/to/agent/key \
                                                -aro agent_id=2
```

You will be prompted for various additional details. For configuration options see, Configuring the PEM server on Linux .

Final steps for PEM Web Application

Perform the following steps on all web application hosts.

1. Open the following ports on the firewall:
    - `8443` for PEM Server (HTTPS)

For example:

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

1. Run the `configure-selinux.sh` script to configure the SELinux policy for PEM.

```
USE_NGINX=1 /usr/edb/pem/bin/configure-selinux.sh
```

1. You can now login to any of the web application instances using the `pemsuper` superuser we created earlier.
2. If you chose to synchronize PEM web application user preferences between instances, complete the setup now by *configuring each web application instance to use the backend for user settings* .

HA PEM using the S1 architecture with EFM and a Virtual IP

This page provides detailed instructions to install and configure a High Availability (HA) PEM deployment according to reference architecture S1.

This example uses EDB Failover Manager (EFM 5.2) for cluster management, EDB Postgres Advanced Server (EPAS 17) as the PEM backend database and Virtual IP (VIP) as the mechanism for routing traffic to the primary on RHEL like systems.

Please see *High availability patterns for PEM deployment* to understand other options.

Also refer to *Deciding on an SSL model* to determine how you will provide SSL certificates and keys prior to starting installation.

- 172.16.161.200 - PEM Backend Primary
- 172.16.161.201 - PEM Backend Standby 1
- 172.16.161.202 - PEM Backend Standby 2
- 172.16.161.203 - EFM Backend Witness Node
- 172.16.161.211 - PEM Web Application 1
- 172.16.161.212 - PEM Web Application 2
- 172.16.161.213 - PEM Web Application 3
- 172.16.161.245 - PEM VIP (used by agents and users to connect)

Deploying an EFM cluster

Before installing PEM, you should have a working EFM cluster on the backend hosts. This section provides instructions to deploy such a cluster. You do not have to follow these steps if you have your own processes to provision EFM clusters. However, if you plan to use sslutils you should ensure it is installed on your cluster before proceeding.

Initial package installation and Postgres configuration

Perform the following steps on all backend nodes unless stated otherwise.

1. Install the following packages:

- EDB Postgres Advanced Server (backend database for PEM Server)
- 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. Initialize a Postgres database and start the service.

```
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. Open the following ports on the firewall:

- `5444` for EPAS
- `7800` for EFM

For example:

```
firewall-cmd --zone=public --add-port=5444/tcp --permanent
firewall-cmd --zone=public --add-port=7800/tcp --permanent
firewall-cmd --reload
```

Optional: install sslutils

If you wish PEM to act as a certificate authority and automatically issue client certificates, you will need to configure *Optional: install sslutils* . If you plan to provide your own certificates, then you may skip this step.

1. Install the sslutils package on all backend hosts.

```
dnf install edb-as17-server-sslutils
```

Set up the primary node for streaming replication

Perform these steps on the primary only.

1. Create the replication role:

```
CREATE ROLE repl REPLICATION LOGIN PASSWORD password;
```

Give the password of your choice.

2. Configure the following in the `postgresql.conf` file:

```
wal_level = replica
max_wal_senders = 10
wal_keep_size = 500
max_replication_slots = 10
```

For more information on configuring parameters for streaming replication, see the PostgreSQL documentation .

3. Add the following entries in the host-based authentication (`/var/lib/edb/as17/data/pg_hba.conf`) file to allow the replication user to connect from all the standbys:

```
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. Restart the EPAS server.

```
systemctl restart edb-as-17.service
```

Add a .pgpass file

Perform these steps on all non-witness backend nodes.

1. Create the `.pgpass` file in the home directory of the enterprisedb user and add the following content. Replace `<password>` with the password of the replication user created previously.

```
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. Set the permissions on the file to restrict access

```
chmod 600 ~/.pgpass
```

Set up the standby nodes for streaming replication

Use the pg_basebackup utility to create replicas of the PEM backend database server on the standby servers.

1. Stop the service for EPAS on all the standby nodes:

```
systemctl stop edb-as-17.service
```

1. Remove the data directory of the database server on all the standby nodes:

```
su - enterprisedb

rm -rf /var/lib/edb/as17/data/*
```

1. Take a backup of the primary node on each of the standby nodes using pg_basebackup:

```
su - enterprisedb -c "\
    /usr/edb/as17/bin/pg_basebackup \
    -h 172.16.161.200 \
    -D /var/lib/edb/as17/data \
    -U repl \
```

(continues on next page)

(continued from previous page)

```
    -v -P -Fp -R -p 5444\
    "
```

1. Start the EPAS database server on each of the standby nodes:

```
systemctl start edb-as-17
```

Set up EFM to manage failover

Perform the following steps to set up EFM:

1. On the primary, create a database user `efm` to connect to the database servers. Grant execute privileges on the functions related to WAL logs, and monitoring privileges, to the user. As a superuser:

```
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;
```

This change will be replicated to the standbys.

1. On the primary, add entries in `pg_hba.conf` to allow the `efm` database user to connect to the database server from all nodes. These entries will be replicated to the standbys.

```
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. Reload the configurations on all the database servers.

```
SELECT pg_reload_conf();
```

4. On all nodes, create an `efm.nodes` file using the sample file (`/etc/edb/efm-5.2/efm.nodes.in`), and give read-write access to the EFM OS user:

```
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. On the standby nodes, add the IP address and EFM port of the primary node in the `/etc/edb/efm-5.2/efm.nodes` file:

6. On all nodes, create the `efm.properties` file using the sample file (`/etc/edb/efm-5.2/efm.properties.in`). Grant read access to all users:

```
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. On any node, encrypt the `efm` database user's password (as defined in Step 1 above) using the `efm` utility, make a note of the output for the next step.

```
export EFMPASS=password
/usr/edb/efm-5.2/bin/efm encrypt efm --from-env
```

8. On all nodes, edit the following parameters in the `efm.properties` file. Replace `<encrypted-password>` with the output of the previous step. Replace `<ip-addr>` with the IP address of each node.

> **Note**
>
> If your hosts are not connected to the internet, replace the value of *ping.server* with the address of a reliable server accessible on your network that will respond to pings.

For more detail on EFM configuration please refer to the documentation . .. ::
edb_notranlate_20

9. On the witness node, set the value of the `is.witness` configuration parameter to `true` :

```
is.witness=true
```

10. On the primary node, enable and start the EFM service:

```
systemctl enable edb-efm-5.2
systemctl start edb-efm-5.2
```

11. On the primary node, allow the standbys to join the cluster:

```
/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. Enable and start the EFM service on the standby nodes and the EFM witness node:

```
systemctl enable edb-efm-5.2
systemctl start edb-efm-5.2
```

13. Check the EFM cluster status by running the following command on any node.

```
/usr/edb/efm-5.2/bin/efm cluster-status efm
```

The output should look like this:

```
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
```

(continues on next page)

(continued from previous page)

```
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.
```

This status confirms that EFM is set up successfully and managing the failover for the PEM server.

In case of failover, any of the standbys are promoted as the primary node, and PEM agents connect to the new primary node. You can replace the failed primary node with a new standby using the procedure above.

Prepare the EFM cluster for PEM installation

Before installing PEM in HA mode, we must ensure that the cluster meets two requirements:

- SSL is configured so that the `cert` auth method can be used for connections from PEM agents to the PEM backend database.
- HBA rules are in place to allow agents and web application instances to connect to the PEM backend.

Prepare SSL files

Follow the instructions in *Generating SSL files for HA PEM backend hosts* .

Modify the EFM cluster to add SSL and HBA rules

Perform the following actions on each PEM backend node (excluding the witness).

1. Modify `postgresql.conf` to enable SSL and set the paths to the SSL files created above.

```
ssl = on
ssl_ca_file = root.crt
ssl_cert_file = server.crt
ssl_key_file = server.key
```

If you have a CRL, also set:

```
ssl_crl_file = root.crl
```

1. Modify `pg_hba.conf` to add the rules required by PEM. Add this block at the top of the file, above any existing content, adjusting the IP addresses to match your cluster.

```
# 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. If using a common agent user, add one additional line to allow the first agent to connect during PEM installation. This line can be removed later if required.

```
hostssl pem      common_agent_user  127.0.0.1/32          cert
```

1. Modify the HBA rules added in the previous sections (for replication and EFM) to `hostssl` instead of `host` to enforce SSL for all connections.

Restart the cluster

> **Note**
>
> By default, EFM will trigger a failover if the database service stops. During this rolling restart, stop the EFM agent before restarting Postgres to prevent an unintended failover. !!!

Perform a rolling restart of the cluster. On each PEM backend host, starting with the standbys and concluding with the primary, do the following:

1. Stop the EFM agent service:

```
systemctl stop edb-efm-5.2
```

1. Restart Postgres:

```
systemctl restart edb-as-17
```

1. Start the EFM agent service:

```
systemctl start edb-efm-5.2
```

Prepare agent users and certificates

If you are using your own SSL certificates, follow these steps to prepare for PEM deployment. If you are using sslutils, you may skip this section.

On the primary, create the common agent user. Grant connect permission on `template1` to this common agent user. This is only required temporarily and can be revoked later.

```
CREATE USER common_agent_user;
GRANT CONNECT ON DATABASE template1 TO common_agent_user;
```

Generate a certificate and key pair for this user and save them on all the backend and web application hosts.

PEM will automatically generate unique, numbered Postgres users (agent1 through agent6) for your deployment across three backend and three frontend hosts.

To support this configuration, you must manually generate six distinct certificate/key pairs—one for each specific agent user—and distribute them so that each host has its own unique pair.

Install PEM on the primary

Perform the following steps on the primary host only.

1. Install the PEM package.

```
dnf install edb-pem --allowerasing
```

1. Create a superuser that can login using a password. This is the user we will use to install PEM. It will also become the agent-server binding user for the agents running on PEM backend hosts.

```
su - enterprisedb -c "psql -c \"create role pemsuper login superuser password your-
↪password-here;\""
```

1. To configure the PEM server, run the configuration script using the Database option (Option 3) and set the host to your Virtual IP (VIP).

To ensure the system is correctly set up for High Availability (HA), you must include the following flags:

- **``--replication-solution efm`` & ``--replication-role primary``** : These flags force the script into HA mode, ensuring compatibility with your failover manager.
- **``--skip-db-certificate``** : This prevents the script from attempting to automatically configure SSL, as you are managing certificates manually in an HA environment.

When using sslutils you must include `--use-existing-sslutils` . This ensures that while the script skips the database SSL setup, it still utilizes the sslutils extension to handle agent certificate generation.

```
/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 \
                                         --replication-role primary \
                                         --skip-db-certificate \
                                         --use-existing-sslutils \
                                         --db-install-path /usr/edb/as17/ \
                                         --db-unitfile edb-as-17
```

When using a common agent user you must supply the username and path to the certificate and key generated for that user.

```
/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 \
                                         --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
```

After running this command, grant the common agent user the pem_agent role and revoke its permissions on `template1` .

```
GRANT pem_agent to common_agent_user;
REVOKE CONNECT ON DATABASE template1 FROM common_agent_user;
```

You may also remove the HBA rule we added for the common agent user at this stage if desired as it will now match the `+pem_agent` rules.

When using your own certificate you should specify the agent_id as `1` and path to the certificate and key generated for the corresponding `agent1` user.

```
/usr/edb/pem/bin/configure-pem-server.sh -t 3 \
                                         -ho 172.16.161.245 \
```

(continues on next page)

(continued from previous page)

```
                                        -p 5444 \
                                        -su pemsuper \
                                        --asb-host-name 127.0.0.1 \
                                        --replication-solution efm \
                                        --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
```

You will be prompted for various additional details. For configuration options see, Configuring the PEM server on Linux .

1. Optionally, to synchronize PEM web application user preferences between instances, *configure central storage of user preferences* . At this stage, you can only complete the backend configuration. We will configure the web application later.

Install PEM on the standbys

Perform the following steps on the standby hosts only.

1. Install the PEM package.

```
dnf install edb-pem --allowerasing
```

1. To configure the PEM server, run the configuration script using the Database option (Option 3) and set the host to your Virtual IP (VIP).

To correctly enable High Availability (HA) mode, you must include the following specific flags:

- ``**--replication-solution efm**`` : Integrates the configuration with Enterprise Failover Manager (EFM).
- ``**--replication-role standby**`` : Designates this node's initial role within the HA cluster.

```
/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 \
                                        --replication-role standby \
                                        --db-unitfile edb-as-17 \
                                        --db-install-path /usr/edb/as17/
```

When using a common agent user you must supply the username and path to the certificate and key generated for that user.

```
/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 \
```

(continues on next page)

(continued from previous page)

```
                                          --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
```

When using your own certificate you should specify the agent_id as the next integer `N` and path to the certificate and key generated for the corresponding `agentN` user.

```
/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 \
                                          --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
```

You will be prompted for various additional details. For configuration options see, Configuring the PEM server on Linux .

Deploy the PEM Web Application

Perform the following steps on all web application hosts.

1. Install the PEM package and the EPAS client binaries:

```
dnf install edb-pem edb-as17-server-client
```

1. Open the following ports on the firewall of all servers:
   - `8443` for PEM Server (HTTPS)

For example:

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

1. Configure the PEM web application. Specify the VIP as the PEM server address.

```
/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/
```

When using a common agent user you must supply the username and path to the certificate and key generated for that user.

```
/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
```

When using your own certificate you should specify the agent_id as the next integer `N` and path to the certificate and key generated for the corresponding `agentN` user.

```
/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
```

You will be prompted for various additional details. For configuration options see, Configuring the PEM server on Linux .

1. Run the `configure-selinux.sh` script to configure the SELinux policy for PEM.

```
USE_NGINX=1 /usr/edb/pem/bin/configure-selinux.sh
```

1. You can now login to any of the web application instances using the `pemsuper` superuser we created earlier.
2. If you chose to synchronize PEM web application user preferences between instances, complete the setup now by *configuring each web application instance to use the backend for user settings* .

Storing user settings in the backend, accessible to all web application instances

When deploying more than one instance of the PEM web application, configure PEM to store user settings in the backend database to ensure they are synchronized across all instances.

Configure the PEM backends

In the PEM backend Postgres instance, connect as a superuser and create a new database.

Grant the new user the right to connect to this database and revoke the right to connect to the PEM database.

```
CREATE DATABASE pem_user_settings;
CREATE USER pem_user_settings WITH PASSWORD choose-a-password;
GRANT CONNECT ON DATABASE pem_user_settings FROM PUBLIC;
REVOKE CONNECT ON DATABASE pem FROM PUBLIC;
```

You should also revoke rights to connect to any other database on this instance, for example `postgres` or `edb` .

You should also ensure that the user `pem_user_settings` is permitted to connect to each Postgres instance from any web application instances. This may require adding a rule to pg_hba.conf of the following form:

```
hostssl  pem_user_settings  pem_user_settings  <web-application-subnet>  scram-sha-256
```

Connect to the new database as a superuser and grant the new user all necessary permissions on the new database.

```
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;
```

Configure the PEM web applications

On each instance of the PEM web application, locate the `config_local.py` file or create one if it does not exist (see Managing configuration settings).

Add the following line, replacing `<password>` with the password of the `pem_user_settings` user.

- If you are using a single endpoint to route traffic to the primary PEM backend instance, then `<host-ip>` and `<host-port>` should be the IP address and port of that endpoint.

```
CONFIG_DATABASE_URI="postgresql://pem_user_settings:<password>@<host-ip>:<host-port>/
↪pem_user_settings"
```

- If you are using multi-host connection strings to route traffic to the PEM primary, you may use the following format, specifying the details of each PEM backend.

```
CONFIG_DATABASE_URI="postgresql://pem_user_settings:<password>@<host-1-ip>:<host-1-
↪port>,<host-2-ip>:<host-2-port>,<host-3-ip>:<host-3-port>/pem_user_settings"
```

HA PEM using the SM architecture with Patroni

This page provides detailed instructions to install and configure a High Availability (HA) PEM deployment according to reference architecture SM.

This example uses Patroni for cluster management, PostgreSQL 17 as the PEM backend database, and multi-host strings as the mechanism for routing traffic to the primary. Our operating system is RHEL 9 x86.

Please see *High availability patterns for PEM deployment* to understand other options.

Also refer to *Deciding on an SSL model* to determine how you will provide SSL certificates and keys prior to starting installation. The examples that follow use these IP addresses:

- 172.16.161.200 - PEM Backend Primary
- 172.16.161.201 - PEM Backend Standby 1
- 172.16.161.202 - PEM Backend Standby 2
- 172.16.161.211 - PEM Web Application 1
- 172.16.161.212 - PEM Web Application 2
- 172.16.161.213 - PEM Web Application 3
- 172.16.161.111 - etcd-node-1
- 172.16.161.112 - etcd-node-2
- 172.16.161.113 - etcd-node-3

Setting up a Distributed Consensus Store with etcd

Patroni requires a Distributed Consensus Store (DCS). If you already have a DCS set up using etcd, you may skip this step and use your existing DCS. Here we will create a minimal three-node DCS on three hosts, separate from the HA PEM application cluster.

On each node:

1. Install etcd from the PGDG repository

```
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. Replace the contents of `/etc/etcd/etcd.conf` with the following, substituting the appropriate values for `<node-ip>` and `node-name` . For example, `172.16.161.111` and `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]
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. Ensure that ports `2380` and `2379` are open on the firewall.

```
firewall-cmd --zone=public --add-port=2380/tcp --permanent
firewall-cmd --zone=public --add-port=2379/tcp --permanent
firewall-cmd --reload
```

4. Ensure that the etcd service is restarted and enabled.

```
systemctl enable etcd
systemctl restart etcd
```

Configuring EDB Repos

On all backend and web application nodes, configure EDB Repos. Refer to Getting started with repositories for more detail.

```
curl -1sLf  https://downloads.enterprisedb.com/<token>/<repo>/setup.rpm.sh | bash
```

Create a Patroni cluster

Before installing PEM, you should have a working Patroni cluster on the backend hosts. This section provides instructions to deploy such a cluster. You do not have to follow these steps if you have your own processes to provision Patroni clusters. However, if you plan to use sslutils you should ensure it is installed on your cluster before proceeding.

Initial package installation and Postgres configuration

Perform the following steps on all backend nodes unless stated otherwise.

1. Install the following packages:

- *Error connecting to PostgreSQL server* (backend database for PEM Server)
- *HA PEM using the SM architecture with Patroni*

```
dnf -qy module disable postgresql
dnf -y install postgresql17-server postgresql17-contrib edb-patroni
```

1. Initialize a Postgres database and start the service.

```
PGSETUP_INITDB_OPTIONS="-E UTF-8" /usr/pgsql-17/bin/postgresql-17-setup initdb
systemctl start postgresql-17
systemctl enable postgresql-17
```

1. Open port `5432` on the firewall:

```
firewall-cmd --zone=public --add-port=5432/tcp --permanent
firewall-cmd --reload
```

Optional: install sslutils

If you wish PEM to act as a certificate authority and automatically issue client certificates, you will need to configure *Optional: install sslutils* . If you plan to provide your own certificates, then you may skip this step.

1. Install the sslutils package on all backend hosts.

```
dnf install sslutils_17
```

Create Patroni users

Perform the following steps on the primary backend host only.

1. Create the Patroni superuser, replication and rewind roles. Execute the following SQL as a superuser.

```
-- 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;
```

Give the passwords of your choice.

Set up Patroni on the primary

1. Create a file at `/etc/patroni/patroni.yml` and add the following content, adjusting the IP addresses to match your cluster and specifying the required cluster and node names.

```
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
        - # 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-
```

(continues on next page)

(continued from previous page)

```
↪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. Stop and disable the Postgres service and start Patroni

```
systemctl disable --now postgresql-17
systemctl enable --now patroni
```

Set up the standby nodes

On each standby node:

1. Stop and disable the service for Postgres.

```
systemctl disable --now postgresql-17
```

2. Remove the data directory of the database server.

```
rm -rf /var/lib/pgsql/17/data
```

3. Create a Patroni configuration file in `/etc/patroni/patroni.yml` . Use the same content as the primary, with the following changes:

- Remove the entire `bootstrap` section
- Change the `name` to a unique name for each standby
- Change the IP addresses under `restapi` and `postgresql.connect_address` to the address of the standby you are configuring.

4. Start and enable Patroni.

```
systemctl enable --now patroni
```

You should now have a functioning Patroni cluster, but without SSL or the specific HBA rules required for PEM. We will configure these in the next section.

Prepare the Patroni cluster for PEM installation

Before installing PEM in HA mode, we must ensure that the cluster meets two requirements:

- SSL is configured so that the `cert` auth method can be used for connections from PEM agents to the PEM backend database.
- HBA rules are in place to allow agents and web application instances to connect to the PEM backend.

Prepare SSL files

Follow the instructions in *Generating SSL files for HA PEM backend hosts* .

Modify the Patroni cluster to add SSL and HBA rules

1. On any backend host, open the editing tool for your cluster using `patronictl edit-config` . For example, the command for the cluster built in the previous section would be:

```
patronictl -c /etc/patroni/patroni.yml edit-config
```

1. Add the following under `postgresql.parameters` :

```
ssl: on
ssl_ca_file: root.crt
ssl_cert_file: server.crt
ssl_key_file: server.key
```

If you have a CRL, also set:

```
ssl_crl_file: root.crl
```

1. Add the following HBA rules to the **top** of `postgresql.pg_hba` above the existing rules, adjusting the IP addresses to match your cluster.

```
- # 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
```

(continues on next page)

(continued from previous page)

```
- 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
- # 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. If using a common agent user, add one additional line to allow the first agent to connect during PEM installation. This line can be removed later if required.

```
- hostssl pem      common_agent_user  127.0.0.1/32          cert
```

1. If you started with a non-SSL cluster, review the existing HBA rules and consider converting `host` rules to `hostssl` rules to force clients to use secure connections if appropriate.

Prepare agent users and certificates

If you are providing your own SSL certificates, follow these steps to prepare for PEM deployment. If you are using sslutils, you may skip this section.

On the primary, create the common agent user. Grant connect permission on `template1` to the common agent user. This is only required temporarily and can be revoked later.

```
CREATE USER common_agent_user;
GRANT CONNECT ON DATABASE template1 TO common_agent_user;
```

Generate a certificate and key pair for this user and save them on all the backend and web application hosts.

PEM will automatically generate unique, numbered Postgres users (agent1 through agent6) for your deployment across three backend and three frontend hosts.

To support this configuration, you must manually generate six distinct certificate/key pairs—one for each specific agent user—and distribute them so that each host has its own unique pair.

Install PEM on the primary

Perform the following steps on the primary host only.

1. Install the PEM package.

```
dnf install edb-pem --allowerasing
```

1. Create a superuser that can login using a password. This is the user we will use to install PEM. It will also become the agent-server binding user for the agents running on PEM backend hosts.

```
su - postgres -c "psql -c \"create role pemsuper login superuser password your-
↪password-here;\""
```

1. To configure the PEM server, run the configuration script using the Database option (Option 3) specifying the IPs of all three backend hosts. Provide the loopback IP address rather than the external IP address for the primary.

To ensure the system is correctly set up for High Availability (HA), you must include the following flags:

- **``--replication-solution efm`` & ``--replication-role primary``** : These flags force the script into HA mode, ensuring compatibility with your failover manager.
- **``--skip-db-certificate``** : This prevents the script from attempting to automatically configure SSL, as you are managing certificates manually in an HA environment.

When using sslutils you must include `--use-existing-sslutils` . This ensures that while the script skips the database SSL setup, it still utilizes the sslutils extension to handle agent certificate generation.

```
/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 \
                                         --replication-role primary \
                                         --skip-db-certificate \
                                         --use-existing-sslutils \
                                         --db-install-path /usr/pgsql-17/ \
                                         --db-unitfile patroni
```

When using a common agent user you must supply the username and path to the certificate and key generated for that user.

```
/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 \
                                         --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
```

After running this command, grant the common agent user the pem_agent role and revoke its permissions on `template1` .

```
GRANT pem_agent to common_agent_user;
REVOKE CONNECT ON DATABASE template1 FROM common_agent_user;
```

You may also remove the HBA rule we added for the common agent user at this stage if desired as it will now match the `+pem_agent` rules.

When using your own certificate you should specify the agent_id as `1` and path to the certificate and key generated for the corresponding `agent1` user.

```
/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 \
                                        --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
```

You will be prompted for various additional details. For configuration options see, Configuring the PEM server on Linux .

Install PEM on the standbys

Perform the following steps on the standby hosts only.

1. Install the PEM package.

```
dnf install edb-pem --allowerasing
```

1. To configure the PEM server, run the configuration script using the Database option (Option 3) and set the host to your Virtual IP (VIP).

To correctly enable High Availability (HA) mode, you must include the following specific flags:

- ``**--replication-solution efm**`` : Integrates the configuration with Enterprise Failover Manager (EFM).
- ``**--replication-role standby**`` : Designates this node's initial role within the HA cluster

```
/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 \
                                        --replication-role standby \
                                        --db-unitfile patroni \
                                        --db-install-path /usr/pgsql-17/
```

When using a common agent user, you must provide the username and path to the certificate and key generated for that user.

```
/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 \
                                        --replication-role standby \
                                        --db-unitfile patroni \
                                        --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
```

When using your own certificate you should specify the agent_id as the next integer `N` and path to the certificate and key generated for the corresponding `agentN` user.

```
/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 \
                                        --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
```

You will be prompted for various additional details. For configuration options see, Configuring the PEM server on Linux .

Deploy the PEM Web Application

Perform the following steps on all web application hosts.

1. Install the PEM package and Postgres (for the client binaries):

```
dnf install edb-pem postgresql17-server
```

1. Open the following ports on the firewall of all servers:
   - `8443` for PEM Server (HTTPS)

For example:

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

1. Configure the PEM web application. Provide a multi-host string including the primary and all standbys as the PEM server address.

```
/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/
```

When using a common agent user you must supply the username and path to the certificate and key generated for that user.

```
/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
```

When using your own certificate you should specify the agent_id as the next integer `N` and path to the certificate and key generated for the corresponding `agentN` user.

```
/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
```

You will be prompted for various additional details. For configuration options see, Configuring the PEM server on Linux .

1. Run the `configure-selinux.sh` script to configure the SELinux policy for PEM.

```
USE_NGINX=1 /usr/edb/pem/bin/configure-selinux.sh
```

1. You can now login to any of the web application instances using the `pemsuper` superuser we created earlier.
2. If you chose to synchronize PEM web application user preferences between instances, complete the setup now by *configuring each web application instance to use the backend for user settings* .

Generating SSL files for HA PEM backend hosts

This page covers how to generate SSL files for an HA PEM backend.

If using an external CA

!!!Note

If you have an existing process for provisioning SSL-enabled Postgres clusters, it is possible that these conditions are already met. In this case no further action required.

Optionally, you may include a Certificate Revocation List (CRL) if one is available.

Ensure these files are owned by the `postgres` or `enterprisedb` user and group, and that the permissions are set to `0600` .

If using sslutils

In HA deployments, PEM does not configure or restart the cluster to avoid clashing with the cluster manager; therefore sslutils must be configured manually.

!!!Note Ensure these steps are carried out as the `postgres` or `enterprisedb` user so the ownership of the files is correct.

1. Generate a single certificate and key pair to be used by sslutils when acting as a certificate authority. Place these in PGDATA on each of the backend hosts.

```
openssl genrsa -out ca_key.key 2048

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. On each backend host, make a copy of `ca_certificate.crt` and name it `root.crt` . This should also be stored in PGDATA.
2. On each backend host, generate a server key, then generate a server certificate signed by the CA. If you plan to use `verify-full` SSL mode for client connections, it is important that the common name matches the DNS name of the server.

```
openssl genrsa -out server.key 2048

openssl req -new -key server.key -out server.csr \
    -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. Set the permissions of the key and certificate files, then remove the CSR.

```
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
```

CHAPTER 8

Installing Postgres Enterprise Manager agent

Select a link to access the applicable installation instructions:

8.1 Linux x86-64 (amd64)

8.1.1 Red Hat Enterprise Linux (RHEL) and derivatives

- *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 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*

8.1.2 SUSE Linux Enterprise (SLES)

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

8.1.3 Debian and derivatives

- *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*

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

8.2.1 Red Hat Enterprise Linux (RHEL) and derivatives

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

8.2.2 SUSE Linux Enterprise (SLES)

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

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

8.3.1 Red Hat Enterprise Linux (RHEL) and derivatives

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

8.3.2 Debian and derivatives

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

8.4 Windows

- *Installing Postgres Enterprise Manager agent on Windows*

8.4.1 Installing Postgres Enterprise Manager agent on Linux x86 (amd64)

Operating system-specific install instructions are described in the corresponding documentation:

Red Hat Enterprise Linux (RHEL) and derivatives

- *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 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 and derivatives

- *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 9 or OL 9 x86_64

Prerequisites

Before you begin the installation process:

- Set up the EDB repository. For more information, see getting started with repositories .

Install the package

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

After installing PEM agent, you need to register the PEM agent. For detailed information see *Registering a PEM agent* .

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

Prerequisites

Before you begin the installation process:

- Set up the EDB repository. For more information, see getting started with repositories .

Install the package

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

After installing PEM agent, you need to register the PEM agent. For detailed information see *Registering a PEM agent* .

Installing Postgres Enterprise Manager agent on AlmaLinux 9 or Rocky

Prerequisites

Before you begin the installation process:

- Set up the EDB repository. For more information, see getting started with repositories .

Install the package

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

After installing PEM agent, you need to register the PEM agent. For detailed information see *Registering a PEM agent* .

Installing Postgres Enterprise Manager agent on AlmaLinux 8 or Rocky

Prerequisites

Before you begin the installation process:

- Set up the EDB repository. For more information, see getting started with repositories .

Install the package

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

After installing PEM agent, you need to register the PEM agent. For detailed information see *Registering a PEM agent* .

Installing Postgres Enterprise Manager agent on SLES 15 x86_64

> **Note**
>
> Postgres Enterprise Manager 8.3 and later is supported on SLES.

Prerequisites

Before you begin the installation process:

- Set up the EDB repository. For more information, see getting started with repositories .
- Activate the required SUSE module:

```
# 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
```

- Refresh the metadata:

```
sudo zypper refresh
```

Install the package

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

After installing PEM agent, you need to register the PEM agent. For detailed information see *Registering a PEM agent* .

Installing Postgres Enterprise Manager agent on Ubuntu 24.04 x86_64

Prerequisites

Before you begin the installation process:

- Set up the EDB repository. For more information, see getting started with repositories .

Install the package

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

After installing PEM agent, you need to register the PEM agent. For detailed information see *Registering a PEM agent* .

Installing Postgres Enterprise Manager agent on Ubuntu 22.04 x86_64

Prerequisites

Before you begin the installation process:

- Set up the EDB repository. For more information, see getting started with repositories .

Install the package

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

After installing PEM agent, you need to register the PEM agent. For detailed information see *Registering a PEM agent* .

Installing Postgres Enterprise Manager agent on Debian 12 x86_64

Prerequisites

Before you begin the installation process:

- Set up the EDB repository. For more information, see getting started with repositories .

Install the package

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

After installing PEM agent, you need to register the PEM agent. For detailed information see *Registering a PEM agent* .

Installing Postgres Enterprise Manager agent on Debian 11 x86_64

Prerequisites

Before you begin the installation process:

- Set up the EDB repository. For more information, see getting started with repositories .

Install the package

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

After installing PEM agent, you need to register the PEM agent. For detailed information see *Registering a PEM agent* .

8.4.2 Installing Postgres Enterprise Manager agent on Linux AArch64 (ARM64)

Operating system-specific install instructions are described in the corresponding documentation:

Red Hat Enterprise Linux (RHEL) and derivatives

- *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 and derivatives

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

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

Prerequisites

Before you begin the installation process:

- Set up the EDB repository. For more information, see getting started with repositories .

Install the package

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

After installing PEM agent, you need to register the PEM agent. For detailed information see *Registering a PEM agent* .

Installing Postgres Enterprise Manager agent on Debian 12 arm64

Prerequisites

Before you begin the installation process:

- Set up the EDB repository. For more information, see getting started with repositories .

Install the package

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

After installing PEM agent, you need to register the PEM agent. For detailed information see *Registering a PEM agent* .

8.4.3 Installing Postgres Enterprise Manager agent on Linux IBM Power (ppc64le)

Operating system-specific install instructions are described in the corresponding documentation:

Red Hat Enterprise Linux (RHEL)

- *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

Prerequisites

Before you begin the installation process:

- Set up the EDB repository. For more information, see getting started with repositories .

Install the package

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

After installing PEM agent, you need to register the PEM agent. For detailed information see *Registering a PEM agent* .

Installing Postgres Enterprise Manager agent on RHEL 8 ppc64le

Prerequisites

Before you begin the installation process:

- Set up the EDB repository. For more information, see getting started with repositories .

Install the package

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

After installing PEM agent, you need to register the PEM agent. For detailed information see *Registering a PEM agent* .

Installing Postgres Enterprise Manager agent on SLES 15 ppc64le

Note

Postgres Enterprise Manager 8.3 and later is supported on SLES.

Prerequisites

Before you begin the installation process:

- Set up the EDB repository. For more information, see getting started with repositories .
- Activate the required SUSE module:

```
# You can use SLES 15 SP3 for PEM 8.3 and later:
sudo SUSEConnect -p PackageHub/15.3/ppc64le

# You can use SLES 15 SP4 for PEM 8.6 and later:
sudo SUSEConnect -p PackageHub/15.4/ppc64le
```

- Refresh the metadata:

```
sudo zypper refresh
```

Install the package

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

After installing PEM agent, you need to register the PEM agent. For detailed information see *Registering a PEM agent* .

8.4.4 Installing Postgres Enterprise Manager agent on Windows

The PEM agent graphical installer for Windows installs and registers the PEM agent.

To invoke the PEM agent installer, assume administrator privileges and navigate to the directory that contains the installer. Then, invoke the installer:

```
pem_agent-9.<x>.<x>-<x>-platform.exe
```

1. The Setup screen opens, welcoming you to the PEM agent installer. Select **Next**.
2. Review the license agreement before selecting the appropriate radio button and accepting the agreement. Select **Next**.
3. By default, the PEM agent is installed in the `C:\Program Files (x86)\edb\pem` directory. You can accept the default installation directory. Or you can modify the contents of the **Installation Directory** field, specifying an alternate installation directory for the PEM agent.

- By default, the PEM agent installer places a certificate in the Administrator's `%APPDATA%\pem` directory. Select **Show advanced options** to include a dialog box in the installation process that allows you to specify an alternate path for the certificate file.
- Select **Register now?** to register the newly installed PEM agent with the PEM server.
- Select **Next**.

1. Enter the connection details for the PEM server in the PEM Server Installation Details dialog box.

- Specify the name or IP address of the system where the PEM database server resides in the **Host** field. If the PEM-HTTPD web server and PEM database are hosted on different systems, you must specify the host of the PEM database.
- Specify the name of the database superuser in the **User Name** field.
- Specify the password associated with the database superuser in the **Password** field.
- Specify the port that PostgreSQL is monitoring in the **Port** field.

1. Select **Next** to continue to pemAgent Service Account. The installer attempts to connect to the server to verify that the details are correct.

> **Note**
>
> The PEM server must allow connections from the PEM agent installer. If you encounter a connection error, confirm the connection properties specified in the PEM Server Installation Details dialog box are correct. Confirm that the *pg_hba.conf* file on the PEM server allows a connection to the server described in the error message.

1. Provide the password for the edb account for the pemAgent service to run under. The agent certificate and key files are created in the `C:\Users\edb\AppData\Roaming\pem` directory. Select **Next**.

2. The tree displayed in the Browser panel of the PEM web interface displays the value entered in the **Description** field to identify the PEM agent. Specify a descriptive name for the agent, such as the hostname of the machine the agent is installed on or a name that reflects the host's functionality.

Provide a descriptive name, or accept the default provided by the PEM Agent host. Select **Next**.

1. If you selected **Show advanced options**, the Advanced Options dialog box opens. By default, the PEM agent installer places the certificate in the `C:\Program Files (x86)\edb\pem` directory. Specify an alternate path for the certificate, or accept the default and select **Next**.
2. The wizard is ready to install the PEM agent. Select **Next**.
3. On the Ready to Install dialog box, select **Next**. The installer copies files to the system and registers the agent on the PEM server.

When the installation completes, the PEM agent is running and reporting operating system and host data to the PEM server. To start monitoring Postgres instances on the host of the PEM agent, add them to PEM's enterprise directory and bound them to the agent.

Invoking the agent installer from the command ine

The command line options of the PEM Agent installers offer functionality in situations where a graphical installation might not work because of limited resources or system configuration. You can:

- Include the `--mode unattended` option when invoking the installer to perform an installation without user input.

Not all command line options are suitable for all platforms. For a complete reference guide to the command line options, include the `--help` option when you invoke the installer.

Invoking the PEM agent installer in unattended mode

You can perform an unattended PEM server installation by providing installation preferences on the command line when invoking the installer. The system where you're installing the PEM server must have internet access.

Before invoking the PEM agent installer in unattended mode, you must:

- Install the PEM server. The `pg_hba.conf` file of the PEM server must allow connections from the host of the PEM agent.
- Ensure that the monitored Postgres database has SSL enabled and is accepting connections.

You need administrator privileges to install the PEM Agent. Use the following command to invoke the PEM agent installer in unattended mode:

```
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>
```

Where:

- `pem_server_host_address` specifies the IP address of the host of the PEM server.
- `pem_server_port` specifies the port used by the backing PEM database. By default, the database uses port 5432.
- `pguser_password` specifies the password associated with the PEM database superuser.
- `agent_name` specifies a descriptive name for the PEM agent.

Note

When configuring a shell/batch script run by a Windows agent, set the *AllowBatchJobSteps* parameter to *True* in the *agent.cfg* file. The PEM agent doesn't execute any batch/shell script by default.

CHAPTER 9

Configuring user authentication in PEM

PEM user authentication is delegated to the Postgres backend server. To be able to login to PEM with a given username, there must exist a Postgres user of that name in the backend. Users must have the `pem_user` role and be permitted by `pg_hba.conf` to connect to the `pem` database from the web application host. For more on PEM user roles see *Managing PEM authentication* .

By default, PEM requires users to authenticate using a password. You can optionally enforce 2FA, or enable password-less authentication using Kerberos.

- *1. Two-factor authentication (2FA)*
- *Kerberos authentication on Linux*
- *Kerberos authentication on Windows*

9.1 Configuring the PEM server to use two-factor authentication

PEM supports two methods for two-factor authentication (2FA):

- Email authentication
- Authenticator app (such as Google Authenticator)

To enable 2FA, you can copy these settings from the `config.py` file to the `config_local.py` file and modify the following parameters.

Parameter	Description
MFA_ENABLED	Set to *true* to enable two-factor authentication. Default value is *false*.
MFA_FORCE_REGISTRATION	Set to *true* to ask the users to register forcefully for the two-factor authentication methods at login. Default value is *false*.
MFA_SUPPORTED_METHODS	Set to *email* to use the email authentication method (send a one-time code by email) or *authenticator* to use the TOTP-based application authentication method.
MFA_EMAIL_SUBJECT	Set to the subject of the email for email authentication. Default value is *<APP_NAME> - Verification Code*.

continues on next page

Table 1 – continued from previous page

Parameter	Description

9.1.1 Mail server configuration

To use the email authentication method, you need to configure mail server settings.

PEM server can send an email using either the SMTP configurations saved in the PEM configuration or using Flask-Mail.

To send the email verification code using the internal SMTP configuration from the PEM configuration, set the parameter `MAIL_USE_PEM_INTERNAL` to `True` . If set to `False` , the following mail configuration is used to send the code to the user-specified email address:

- MAIL_SERVER = 'localhost'
- MAIL_PORT = 25
- MAIL_USE_TLS = False
- MAIL_USE_SSL = False
- MAIL_USERNAME = None
- MAIL_PASSWORD = None
- MAIL_DEFAULT_SENDER = None

For more details about these configurations, see the Flask-Mail documentation .

Note

PEM SMTP alerts don't use this configuration.

9.2 Configuring the PEM server to use Kerberos authentication

You can configure Kerberos authentication for the PEM server. The Kerberos server works with hostnames and not with IP addresses. To use single sign-on in PEM server using Kerberos authentication, configure the following machines with hostnames using the DNS (realm).

- Kerberos server
- PEM server (PEM web server and PEM backend database server)
- Client machine

For example, if the realm on Kerberos server is `edbpem.org` , then you can set the Kerberos server hostname to `Krb5server.edbpem.org` , the PEM server hostname to `pem.edbpem.org` , and the client's hostname to `pg12.edbpem.org` . The convention is to use the DNS domain name as the name of the realm.

9.2.1 1. Install Kerberos, the PEM server, and the PEM backend database

Install Kerberos on the machine that functions as the authentication server. Install the PEM server on a separate machine. For more information, see *Installing the PEM server and PEM-HTTPD on separate hosts* .

Install the PEM backend database (Postgres/EDB Postgres Advanced Server) on the same machine as the PEM server or on a different one. For more information, see the installation steps on EDB Docs website .

9.2.2 2. Add principals on Kerberos server

Add the principals for the PEM web application deployed under an Apache web server (HTTPD/Apache2) and the PEM backend database server (PostgreSQL/EDB Postgres Advanced Server).

```
$ 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` must contain the realm of the Kerberos server. For example, you can specify `pemdb.edbpem.org` as the hostname of PEM server, with `edbpem.org` as the realm.

Note

If the PEM web application and the PEM backend database server are on different machines, then hostname is different for each one.

9.2.3 3. Extract key tables from Kerberos server

Extract the key tables from Kerberos for the PEM web application and the PEM backend database server:

```
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>"
```

Copy the key tables from the Kerberos server to the PEM server:

```
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
```

On the PEM server, move the key tables to the required location and change ownership:

```
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
```

Where:

- `NAME_OF_PEM_WEB_FILE` is the name specified for the key table for the PEM web application.
- `NAME_OF_PEM_DB_FILE` is the name specified for the key table for the PEM backend database server.
- `OS_USERNAME_ON_PEM_SERVER` is the name of the operating system user on the PEM server.
- `DATA_DIRECTORY_OF_POSTGRES` is the path of the data directory of the installed Postgres database (PostgreSQL/EDB Postgres Advanced Server).

9.2.4 4. Configure the PEM backend database server

Add the key table location in the `postgresql.conf` file:

```
krb_server_keyfile=FILE:/<DATA_DIRECTORY_OF_POSTGRES>/<NAME_OF_PEM_DB_FILE>.keytab
```

Where:

- NAME_OF_PEM_DB_FILE is the name specified for the key table for the PEM backend database server.
- DATA_DIRECTORY_OF_POSTGRES is the path of the data directory of the installed Postgres database (PostgreSQL/EDB Postgres Advanced Server).

Edit the krb5.conf file:

```
$ 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
}
```

Restart the database server to reflect the changes:

```
systemctl restart <POSTGRES_SERVICE_NAME>
```

POSTGRES_SERVICE_NAME is the service name of the Postgres (PostgreSQL/EDB Postgres Advanced Server) database, for example, postgresql-13 for PostgreSQL 13 database on a RHEL or Rocky Linux platform.

9.2.5 5. Obtain and view the initial ticket

The kinit utility obtains and caches Kerberos tickets. You typically use this utility to obtain the ticket-granting ticket, entering a password to decrypt the credential from the key distribution center (KDC). The ticket-granting ticket is then stored in your credential cache.

You can view the details of the ticket using the klist utility.

Note

Install the Kerberos client on the PEM server and the client machine before using kinit and klist.

```
$ kinit <USERNAME@REALM>
$ klist
```

These commands display the principal along with the Kerberos ticket.

Note

The *USERNAME@REALM* specified here must be a database user having the pem_admin role and CONNECT privilege on the *pem* database.

9.2.6 6. Configure the PEM server

Run the PEM configure script on the PEM server to use Kerberos authentication:

```
$ 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
```

Configure `PEM_DB_HOST` in the `config_setup.py` file. Check that the value of `PEM_AUTH_METHOD` is set to `'kerberos'`.

```
$ sudo vim <PEM_INSTALLATION_DIRECTORY>/share/web/config_setup.py
PEM_DB_HOST=`<HOSTNAME_OF_PEM_SERVER>`
```

Configure the host in the `.install-config` file:

```
$ sudo vim <PEM_INSTALLATION_DIRECTORY>/share/.install-config
HOST=`<HOSTNAME_OF_PEM_SERVER>`
```

If the PEM server uses Kerberos authentication:

- All the monitored servers default to use the same authentication. To override the default, in the `config_local.py` file, add the parameter `ALLOW_DATABASE_CONNECTION_WITHOUT_KERBEROS` and set it to `True`.
- All the authenticated user principals are appended with the realm (USERNAME@REALM) and passed as the database user name by default. To override the default, in the `config_local.py` file, add the parameter `PEM_USER_KRB_INCLUDE_REALM` and set it to `False`.
- Restart the Apache server:

```
sudo systemctl restart <SERVICE_NAME>
```

- Edit the entries at the top of `pg_hba.conf` to use the gss authentication method, and reload the database server:

```
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` is the service name of the Postgres (PostgreSQL/EDB Postgres Advanced Server) database, for example, `postgresql-13` for PostgreSQL 13 database on a `RHEL` or Rocky Linux platforms.

Note

If you're using PostgreSQL or EDB Postgres Advanced Server 12 or later, you can specify the connection type as *hostgssenc* to allow only gss-encrypted connections.

9.2.7 7. Browser settings

Configure the browser on the client machine to access the PEM web client to use Spnego/Kerberos.

For Mozilla Firefox:

1. Open the low-level Firefox configuration page by loading the `about:config` page.
2. In the search box, enter `network.negotiate-auth.trusted-uris`.

3. Double-click the `network.negotiate-auth.trusted-uris` preference and enter the hostname or the domain of the web server that's protected by Kerberos HTTP SPNEGO. Separate multiple domains and hostnames with commas.
4. In the search box, enter `network.negotiate-auth.delegation-uris` .
5. Double-click the `network.negotiate-auth.delegation-uris` preference and enter the hostname or the domain of the web server that's protected by Kerberos HTTP SPNEGO. Separate multiple domains and hostnames with commas.
6. Select **OK**.

For Google Chrome on Linux or MacOS:

- Add the `--auth-server-whitelist` parameter to the `google-chrome` command. For example, to run Chrome from a Linux prompt, use this `google-chrome` command:

```
google-chrome --auth-server-whitelist = "hostname/domain"
```

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

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

Note

You might see the following error while connecting to your Postgres cluster:

```
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
```

Add encryption types to the keytab using ktutil or by re-creating the Postgres keytab with all crypto systems from AD.

9.3 Configuring the PEM server to use Windows Active Directory domain services for Kerberos authentication (SSPI)

The Windows Active Directory domain service works with hostnames and not with IP addresses. To use single sign-on in PEM server using Active Directory domain services, configure the following machines with hostnames using the DNS:

- Windows server (domain controller)
- PEM server (PEM web server and PEM backend database server)
- Client machine

For example, if the realm on Windows Active Directory is `edbpem.internal` , then you can set the Windows server hostname to `Krb5server.edbpem.internal` , the PEM server hostname to `pem.edbpem.internal` , and the client's hostname to `pg12.edbpem.internal` .

9.3.1 1. Install Active Directory, the PEM server, and the PEM backend database server

Perform the following installations:

- Install Active Directory on the Windows server (domain controller) that functions as the authentication server. Also, configure the Active Directory domain services to use Kerberos authentication, and then start it.

- Install the PEM server on a separate Linux machine. For more information, see *Installing the PEM server and PEM-HTTPD on separate hosts* .
- Install the PEM backend database (Postgres/EDB Postgres Advanced Server) on the same Linux machine as the PEM server or a different one. For more information, see the installation steps on the EDB Docs website .

9.3.2 2. Create users in Active Directory to map with service principals

Create users in Active Directory of the Windows server to map with the HTTP service principal for the PEM web application.

1. Open **Active Directory Users and Computers > <DOMAIN_NAME> > Users**. Right-click and select **New > User**.

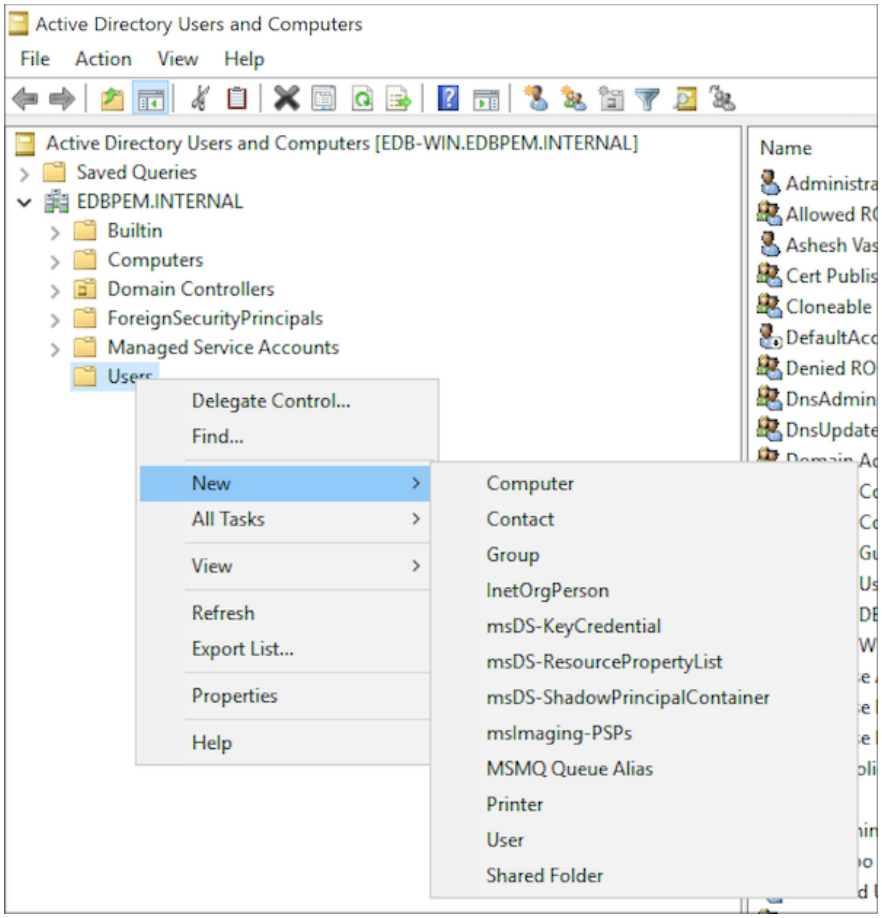


Fig. 1: Active Directory Users and Computers

1. Enter the user details.
2. Enter the password and make sure to clear **User must change password at next logon**. Also select **User cannot change password** and **Password never expires**.
3. Review the user details.
4. On the PEMServer Web Properties dialog box, add the users as members of the Domain Users group:

1. Create the user (for example, pemserverdb) in Active Directory on the Windows server to map with the Postgres service principal for the PEM backend database.

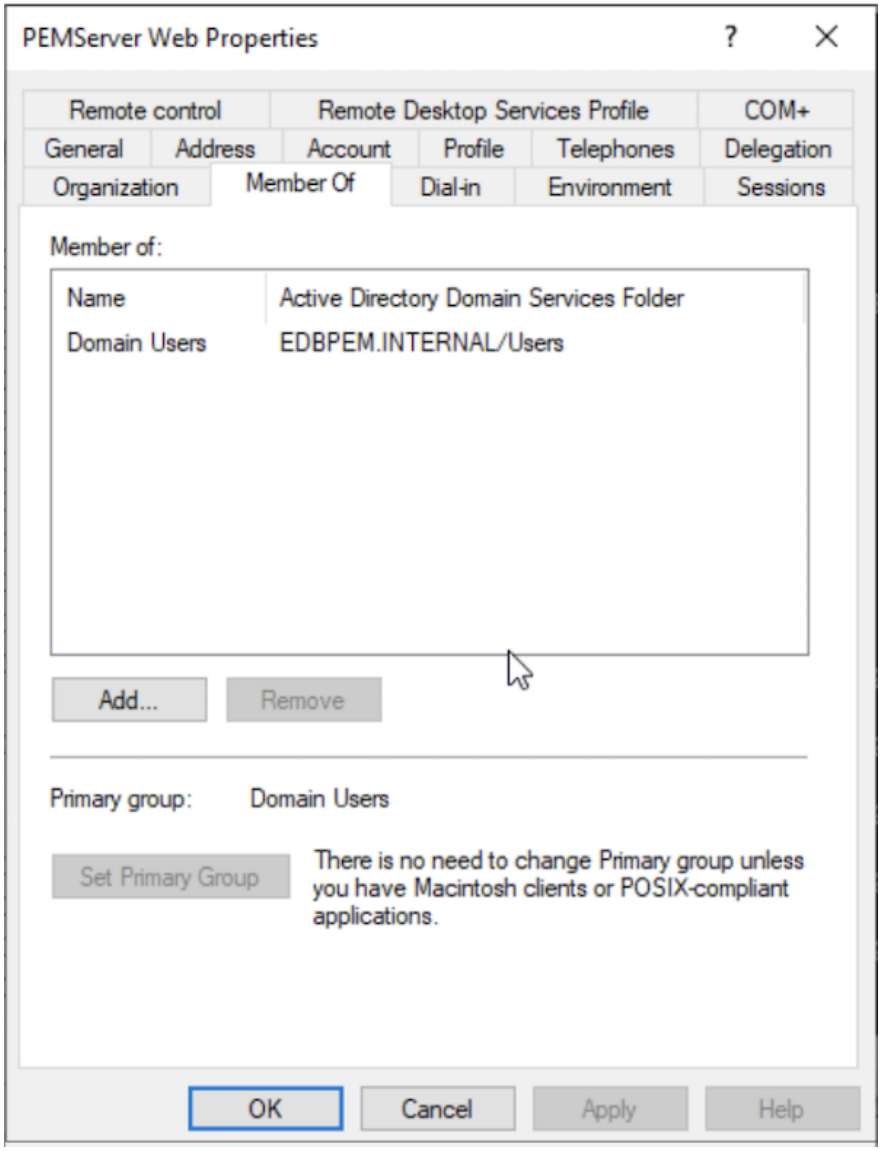


Fig. 2: PEM Server Web Properties

9.3.3 3. Extract key tables from Active Directory

Extract the key tables for the service principals and map them with the respective domain users you created.

1. Open Windows PowerShell as an administrator. Create a key table for HTTP service principal mapping with the user pemserverweb and a key table for Postgres service principal mapping with the user pemserverdb:

```
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
```

Where:

```
- `pemserver.keytab` is the name of the key table for the PEM web application. - ␣
↪`pemdb.keytab` is the name of the key table for the PEM backend database server. -␣
↪ `pem.edbpem.internal@EDBPEM.INTERNAL` is the hostname of the PEM server. Here,␣
↪@EDBPEM.INTERNAL means @REALM. - `pemserverweb` is the user for the PEM web␣
↪application. - `pemserverdb` is the user for the PEM backend database server. - ␣
↪ `EDBPEM.INTERNAL` is the domain of the target.
```

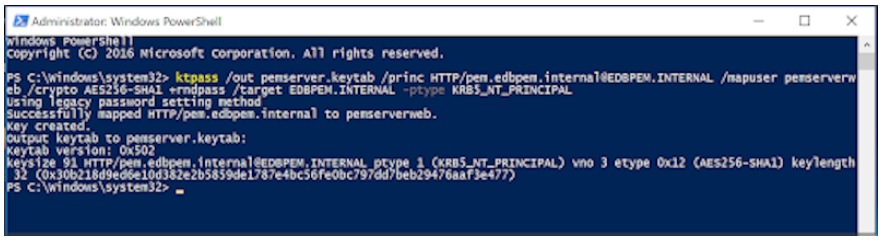


Fig. 3: Windows PowerShell ktpass

Note

The command line argument *+rndpass* resets the password for the domain user pemserverweb to a random password. The */target* option is optional.

1. On the **Accounts** tab, add Kerberos support for the user accounts.

Note

On the **Accounts** tab, the user logon name shows **HTTP/pem.edbpem.internal@EDBPEM.INTERNAL**. The **Delegation** tab is enabled for the pemserverweb user.

1. On the **Delegation** tab, select **Trust this user for delegation to any service (Kerberos only)** for the users you created.
2. Copy both the key tables to the PEM server host or to the PEM web server and PEM backend database server hosts if installed on different hosts.
3. On the PEM server, move the key tables to the required location and change the ownership:

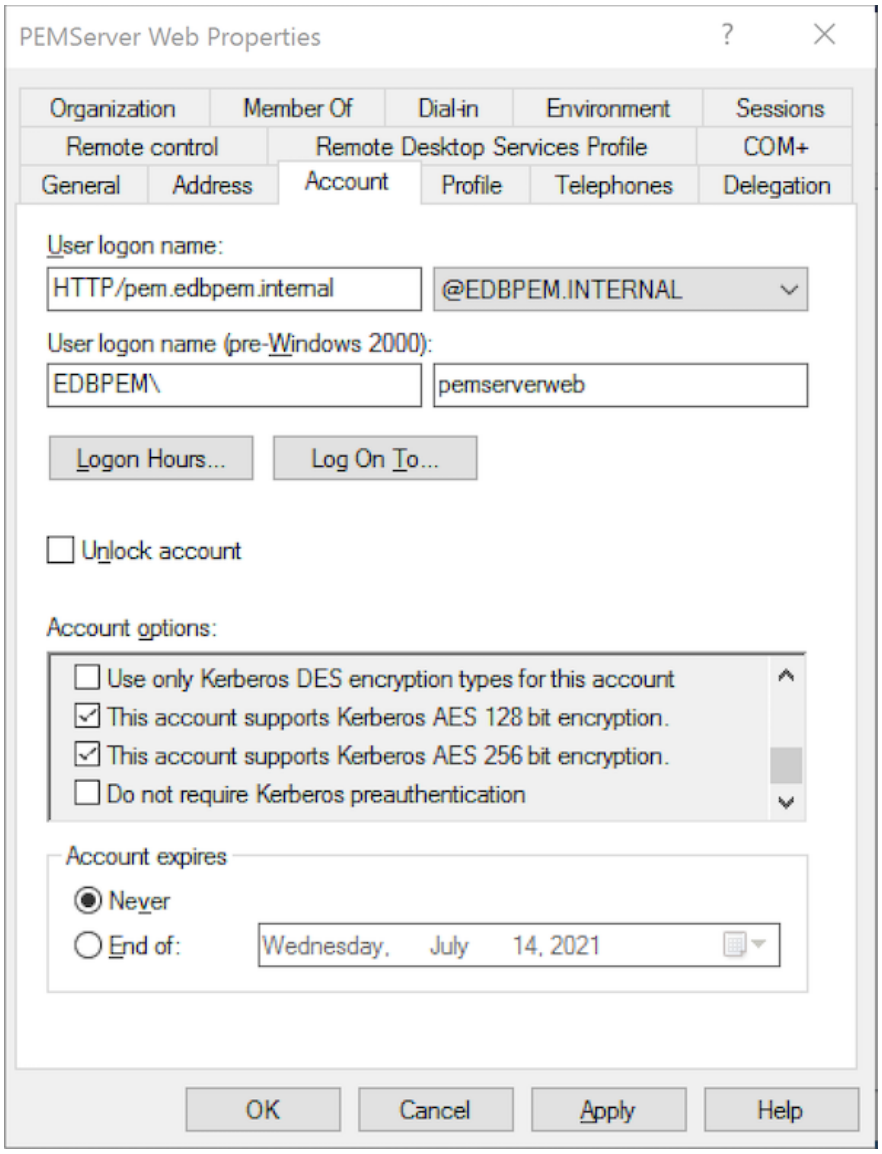


Fig. 4: PEM Server Web Properties

```
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
```

Where:

```
- `OS_USERNAME_ON_PEM_SERVER` is the name of the operating system user on the PEM␣
↪server. - `DATA_DIRECTORY_OF_POSTGRES` is the path of the data directory of the␣
↪installed Postgres database (PostgreSQL/EDB Postgres Advanced Server).
```

9.3.4 4. Configure the PEM backend database server

Add the key table location in the `postgresql.conf` file:

```
krb_server_keyfile=FILE:/<DATA_DIRECTORY_OF_POSTGRES>/pemdb.keytab
```

Where `DATA_DIRECTORY_OF_POSTGRES` is the path of the data directory of the installed Postgres database (PostgreSQL/EDB Postgres Advanced Server).

Edit the `krb5.conf` file:

```
$ 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
}
```

Restart the database server to reflect the changes:

```
systemctl restart <POSTGRES_SERVICE_NAME>
```

`POSTGRES_SERVICE_NAME` is the service name of the Postgres (PostgreSQL/EDB Postgres Advanced Server) database, for example, `postgresql-13` for PostgreSQL 13 database on RHEL or Rocky Linux platforms.

9.3.5 5. Obtain and view the initial ticket

The kinit utility obtains and caches Kerberos tickets. You typically use this utility to obtain the ticket-granting ticket, using a password you entered to decrypt the credential from the key distribution center (KDC). The ticket-granting ticket is then stored in your credential cache.

You can view the details of the ticket using the klist utility.

> **Note**
>
> You must install the Kerberos client on the PEM server and the client machine before using kinit and klist.

```
$ kinit <USERNAME@REALM>
$ klist
```

These commands display the principal along with the Kerberos ticket.

> **Note**
>
> The *USERNAME@REALM* specified here must be a database user having the pem_admin role and CONNECT privileges on the *pem* database.

9.3.6 6. Configure the PEM server

Run the PEM configure script on the PEM server to use Kerberos authentication:

```
$ sudo PEM_APP_HOST=pem.edbpem.internal PEM_KRB_KTNAME=<PEM_INSTALLATION_DIRECTORY/
↪share/pemserver.keytab <PEM_INSTALLATION_DIRECTORY>/bin/configure-pem-server.sh
```

In the `config_setup.py` file, configure `PEM_DB_HOST` and check that the value of `PEM_AUTH_METHOD` is set to `'kerberos'`:

```
$ sudo vim <PEM_INSTALLATION_DIRECTORY>/share/web/config_setup.py
PEM_DB_HOST=`pem.edbpem.internal`
```

Configure `HOST` in the `.install-config` file:

```
$ sudo vim <PEM_INSTALLATION_DIRECTORY>/share/.install-config
HOST=`pem.edbpem.internal`
```

If the PEM server uses Kerberos authentication:

- All the monitored servers default to use the same authentication. To override the default, in the `config_local.py` file, add the parameter `ALLOW_DATABASE_CONNECTION_WITHOUT_KERBEROS` and set it to `True`.
- All the authenticated user principals are appended with the realm (USERNAME@REALM) and passed as the database user name by default. To override the default, in the `config_local.py` file, add the parameter `PEM_USER_KRB_INCLUDE_REALM` and set it to `False`.

Restart the Apache server:

```
sudo systemctl restart <SERVICE_NAME>
```

Edit the entries at the top in `pg_hba.conf` to use the gss authentication method. Then reload the database server:

```
=            shell 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` is the service name of the Postgres (PostgreSQL/EDB Postgres Advanced Server) database, for example, `postgresql-13` for PostgreSQL 13 database on RHEL or Rocky Linux platforms.

Note

You can't specify the connection type as *hostgssenc* . Windows doesn't support gss-encrypted connections.

9.3.7 7. Browser settings

Configure the browser on the client machine to access the PEM web client to use Spnego/Kerberos.

For Mozilla Firefox:

- Open the low-level Firefox configuration page by loading the `about:config` page.
- In the search box, enter `network.negotiate-auth.trusted-uris` .
- Double-click the `network.negotiate-auth.trusted-uris` preference and enter the hostname or the domain of the web server that's protected by Kerberos HTTP SPNEGO. Separate multiple domains and hostnames with commas.
- In the search box, enter `network.negotiate-auth.delegation-uris` .
- Double-click the `network.negotiate-auth.delegation-uris` preference and enter the hostname or the domain of the web server that's protected by Kerberos HTTP SPNEGO. Separate multiple domains and hostnames with commas.
- Select **OK**.

For Google Chrome on Linux or MacOS:

- Add the `--auth-server-whitelist` parameter to the `google-chrome` command. For example, to run Chrome from a Linux prompt, use this `google-chrome` command:

```
google-chrome --auth-server-whitelist = "hostname/domain"
```

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

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

Note

You might see the following error while connecting to your Postgres cluster:

```
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
```

Add encryption types to the keytab using ktutil or by re-creating the Postgres keytab with all crypto systems from AD.

CHAPTER 10

Connection pooling using PgBouncer

To limit connections between the PEM agent and server on non-Windows systems, you can implement PgBouncer as a connection pooler. The process involves three main phases:

- ** *Preparing the PEM database server* **

It provides the steps to prepare the PEM database server to use with PgBouncer.

- ** *Configuring PgBouncer* **

It provides detailed information about configuring PgBouncer to allow it to work with the PEM database server.

- ** *Configuring the PEM agent* **

It provides detailed information about configuring a PEM agent to connect to PgBouncer.

For detailed information about using the PEM web interface, see *Accessing the web interface* .

10.1 Preparing the PEM database server

To enable connection pooling for PEM with PgBouncer, you must configure dedicated users and create an SSL key and certificate on the PEM database server.

This example shows how to prepare the PEM database server with the enterprisedb user on a RHEL-based operating system with EDB Postgres Advanced Server version 16. The location of your data, the configuration and key files, and the user you use to perform the configuration may differ depending on your OS and Postgres distribution.

10.1.1 Prerequisites

- Connect to the `pem` database of the PEM database server.
- Connect as the enterprisedb or postgres user based on your Postgres distribution.

Postgres distribution	User
EDB Postgres Advanced Server	enterprisedb
EDB Postgres Extended Server	postgres
PostgreSQL	postgres

10.1.2 Creating users and roles for PgBouncer-PEM connections

1. Create a dedicated user named pgbouncer with pem_agent_pool membership. This user will serve connections from PgBouncer to the PEM database by forwarding all agent database queries.

```
CREATE ROLE pgbouncer PASSWORD ANY_PASSWORD LOGIN;
__OUTPUT__
CREATE ROLE
```

```
GRANT pem_agent_pool TO pgbouncer;
__OUTPUT__
GRANT ROLE
```

1. Create a user named pem_admin1 (not a superuser) with pem_admin and pem_agent_pool role membership. This user registers the agent to the PEM server and manages access to the PEM database.

```
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. Grant CONNECT privileges to the pgbouncer user:

```
GRANT CONNECT ON DATABASE pem TO pgbouncer;
__OUTPUT__
GRANT
```

1. Grant USAGE privileges to the pgbouncer user for the `pem` schema:

```
GRANT USAGE ON SCHEMA pem TO pgbouncer;
__OUTPUT__
GRANT
```

1. Grant EXECUTE privileges to the pgbouncer user on the `pem.get_agent_pool_auth(text)` function. For example:

```
GRANT EXECUTE ON FUNCTION pem.get_agent_pool_auth(text) TO pgbouncer;
__OUTPUT__
GRANT
```

1. Use the `pem.create_proxy_agent_user(varchar)` function to create a user named pem_agent_user1. This proxy user will serve connections between all agents and PgBouncer.

```
SELECT pem.create_proxy_agent_user(pem_agent_user1);
__OUTPUT__
create_proxy_agent_user
-------------------------
(1 row)
```

The function creates a user with the same name and a random password and grants pem_agent and pem_agent_pool roles to the user. This approach allows PgBouncer to use a proxy user on behalf of the agent.

10.1.3 Updating the configuration files to allow PgBouncer-PEM connections

1. Allow the pgbouncer user to connect to the `pem` database using the SSL authentication method. To do so, add the `hostssl pem` entry in the `pg_hba.conf` file of the PEM database server.

In the list of rules, be sure to place the `hostssl pem` entry before any other rules assigned to the +pem_agent user.

```
# 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. Allow the PEM server to map all users involved in PgBouncer-PEM connections by adding these lines to the `$PGDATA/pg_ident.conf` user mapping file:

```
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. Restart the Postgres service. Replace `<postgres_service>` with the name of the Postgres instance systemd service name:

```
systemctl restart <postgres_service>
```

10.1.4 Creating the SSL key and certificate for PgBouncer-PEM authentication

Create a key and certificate for the pem_agent_pool group role. Then, move the files to the PgBouncer instance to allow authentication between the PEM database server and PgBouncer.

This example runs EDB Postgres Advanced Server on RHEL. When setting your environment variables, choose the correct directories according to your operating system and Postgres distribution.

1. Set the `$DATA_DIR` environment variable to your data directory:

```
export DATA_DIR=/var/lib/edb/as16/data
```

Data directories per OS and Postgres version

Here are some examples of other default data directories per operating system and Postgres version.

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. Set the `$USER_HOME` environment variable to the home directory accessible to the user:

```
export USER_HOME=/var/lib/edb
```

User home directories per OS and Postgres version

Here are some examples of other default home directories per operating system and Postgres version.

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. Create the signing key with openssl:

```
openssl genrsa -out pem_agent_pool.key 4096
```

1. Create a certificate-signing request (CSR). Replace the `-subj` attributes in `<...>` as required. Ensure the common name (CN) is set to the pem_agent_pool group role name:

```
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. Use the PEM CA and key to sign the CSR:

```
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. Move the created key and certificate to a path the enterprisedb user can access.

In this example, create a folder `~/.postgresql` in the home directory of the enterprisedb user and ensure it has permissions:

```
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.*
```

10.2 Configuring PgBouncer

You must configure PgBouncer to work with the PEM database server.

10.2.1 Prerequisites

- If you're running EDB Postgres Advanced Server, install EDB PgBouncer .
- If you're running EDB Postgres Extended Server or PostgreSQL, install community PgBouncer .

EDB PgBouncer and PgBouncer installation considerations

The name and location of the directories and files in the configuration steps, as well as the user, depend on whether you installed the community version of PgBouncer or EDB PgBouncer. If you installed community PgBouncer (whether you install it from the community repo or the EDB repo), replace the names of the files and directories in the worked example with the values for PgBouncer.

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*

10.2.2 Configuring PgBouncer

This example configures EDB PgBouncer with the enterprisedb system user.

If you're running community PgBouncer, replace the names of the directories, files, and user as explained in *EDB PgBouncer and PgBouncer installation considerations* .

1. Open a terminal window and navigate to the PgBouncer directory.
2. Change the owner of the `etc` directory for PgBouncer (where `edb-pgbouncer.ini` resides) to `enterprisedb` , and change the directory permissions to `0700` :

```
chown -R enterprisedb:enterprisedb /etc/edb/pgbouncer<1.x>
chmod 0700 /etc/edb/pgbouncer<1.x>
```

1. Change the contents of the `edb-pgbouncer.ini` file:

```
  [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
  ;; for PEM Agents
  client_tls_sslmode = require
```

(continues on next page)

(continued from previous page)

```
;; 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 For more information on *auth_user* , see Authentication settings . .. ::

1. Create an HBA file (*/etc/edb/pgbouncer<1.x>/hba_file*) for PgBouncer that contains the following content:

```
# 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
hostssl pem pem_agent_user1 0.0.0.0/0 cert
```

1. Change the owner of the HBA file `(/etc/edb/pgbouncer<1.x>/hba_file)` to `enterprisedb` , and change the directory permissions to `0600` :

```
chown enterprisedb:enterprisedb /etc/edb/pgbouncer<1.x>/hba_file
chmod 0600 /etc/edb/pgbouncer<1.x>/hba_file
```

1. Enable the PgBouncer service, and start the service:

```
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>
```

10.3 Configuring the PEM agent

10.3.1 Prerequisites

Install the PEM agent .

!!!note Don't configure PEM agents with `enable_smtp` , `enable_snmp` , or `enable_webhook` set to `true` in the `agent.cfg` file to connect through PgBouncer. SNMP, SMTP, and Webhook spoolers use the LISTEN/NOTIFY mechanism provided by Postgres to send notifications asynchronously. Since PgBouncer doesn't support the LISTEN/NOTIFY mechanism in transaction mode, connecting the agent to PgBouncer can cause notifications to be delayed or not delivered at all. Instead, connect the PEM agent directly to the PEM backend database.

10.3.2 Configuring a new PEM agent

After installing the PEM agent, configure it to work with a particular PEM database server:

```
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!
```

In this command, the `--pem-agent-user` argument instructs the agent to create an SSL certificate and key pair for the pem_agent_user1 database user in the `/root/.pem` directory.

For example:

```
/root/.pem/pem_agent_user1.crt

/root/.pem/pem_agent_user1.key
```

The PEM agent uses the keys to connect to the PEM database server as pem_agent_user1. It also creates an agent configuration file named `/usr/edb/pem/agent/etc/agent.cfg` .

A line mentioning the agent-user to use appears in the `agent.cfg` configuration file. For example:

```
cat /usr/edb/pem/agent/etc/agent.cfg
[PEM/agent]
pem_host=172.16.254.22
pem_port=6432
```

(continues on next page)

(continued from previous page)

```
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
```

10.3.3 Configuring an existing PEM agent

If you're using an existing PEM agent, you can copy the SSL certificate and key files to the target machine and reuse the files. You must modify the files, adding a new parameter and replacing some parameters in the existing `agent.cfg` file.

1. Add a line to use agent_user as the agent:

```
agent_user=pem_agent_user1
```

1. Update the port to specify the PgBouncer port:

```
pem_port=6432
```

1. Update the certificate and key path locations:

```
agent_ssl_key=/root/.pem/pem_agent_user1.key
agent_ssl_crt=/root/.pem/pem_agent_user1.crt
```

As an alternative, you can run the agent self-registration script. However, that process creates a new agent id. If you run the agent self-registration script, you must replace the new agent id with the existing id and disable the entry for the new agent id in the `pem.agent` table. For example:

```
pem=# UPDATE pem.agent SET active = false WHERE id = <new_agent_id>;
__OUTPUT__

UPDATE 1
```

Note

Keep a backup of the existing SSL certificate, key file, and agent configuration file.

CHAPTER 11

Upgrade and migration

You can upgrade the Postgres Enterprise Manager (PEM) components including the PEM installation, the backend database, and the SQL Profiler. You can also move the PEM server.

11.1 Upgrading a PEM installation

The process of upgrading a PEM installation is platform specific. You can update a PEM agent or server on a Linux platform by using the native packages and on a Windows platform by using the PEM graphical installer available for Windows.

Links to PEM installers and RPMs are available at the EDB website .

11.1.1 Upgrading a PEM installation on a Linux host

To upgrade PEM component software on Linux hosts, install a newer version of the PEM component packages in the following order:

1. Invoke the PEM agent package installer on each monitored node except the PEM server host.
2. Invoke the PEM server package installer. It upgrades both the PEM server and the PEM agent that resides on the PEM server host. If the web application and backend database are running on separate hosts, update the database host first.

During an installation, the component installation detects an existing installation and performs an upgrade. After upgrading the PEM agent and server, you can upgrade SQL Profiler if required. That step is platform specific.

Upgrading a PEM server installation

If you want to upgrade a PEM server that's installed on a machine in an isolated network, you need to create a PEM repository on that machine before you upgrade the PEM server. For more information, see *Creating an EDB repository on an isolated network* .

To upgrade a PEM server installation:

```
sudo <package-manager> upgrade edb-pem
```

Where `<package-manager>` is the package manager used with your operating system:

Package manager	Operating system
dnf	RHEL 8/9 and derivatives
zypper	SLES
apt-get	Debian and Ubuntu

If upgrading from PEM 9 on a RHEL-like system, you may need to include the `--allowerasing` option to prevent the upgrade from being blocked by the obsolete edb-pem-docs package.

```
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] ?
```

```
- To perform an unattended upgrade, you can use `apt`  options to force a decision: -␣
↪**To install the new package version:** Add `o Dpkg::Options::="--force-confnew"` ␣
↪to your command. - Example: `apt -o Dpkg::Options::="--force-confnew" install edb-
↪pem`  - **To keep your existing version:** Add `o Dpkg::Options::="--force-confdef"␣
↪-o Dpkg::Options::="--force-confold"`  to your command. - Example: `apt -o␣
↪Dpkg::Options::="--force-confdef" -o Dpkg::Options::="--force-confold" install edb-
↪pem`  After upgrading the PEM server, you must configure the PEM server. For␣
↪detailed information, see  :ref:`Configuring the PEM server to use two-factor␣
↪authentication <Configuring the PEM server to use two-factor authentication>`  .
```

Note

If you upgrade the PEM backend database server and the PEM server, update the *PG_INSTALL_PATH* and *DB_UNIT_FILE* parameters pointing to the new version in the */usr/edb/pem/share/.install-config* file before you run the configure script.

Configuring the PEM server

After upgrading the PEM server, you can use the following command to configure the PEM server:

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

The configure script uses the values from the old PEM server configuration file while running the script. For details, see

Configuring the PEM server on Linux platforms .

Note

- The configure script requires a superuser password only after the upgrade process.

```
- If your configure script gets stuck, then stop the PEM agent with `alert_threads>0`␣
↪. To get the details of such agents, execute the query:
```

```
SELECT agent_id FROM pem.agent_config WHERE param=alert_threads AND value > 0;
```

Stop the running agents and rerun the configure script.

If the problem persists, then run the query to terminate the stuck alert processes:

```
SELECT pg_terminate_backend(pid) FROM pg_stat_activity WHERE query=SELECT pem.process_
↪one_alert();
```

Then rerun the configure script.

Upgrading a PEM agent installation

To upgrade a PEM agent:

```
sudo <package-manager> upgrade edb-pem-agent
```

Where `<package-manager>` is the package manager used with your operating system:

Package manager	Operating system
dnf	RHEL 8/9 and derivatives
zypper	SLES
apt-get	Debian and Ubuntu

11.1.2 Upgrading a PEM installation on a Windows host

To upgrade PEM component software on Windows hosts, invoke a newer version of the PEM component installers in the following order:

1. Invoke the PEM agent installer on each monitored node except the PEM server host.
2. Invoke the PEM server installer. This installer upgrades both the PEM server and the PEM agent that resides on the PEM server host.

During an installation, the component installer automatically detects an existing installation and performs an upgrade. After upgrading the PEM gent and server, you can upgrade SQL profiler if required. This step is platform specific.

Upgrading a PEM agent on a Windows host

To upgrade a system that is currently monitored by a PEM agent to a more recent PEM agent, download and invoke a newer version of the PEM Agent installer on the system that the agent is monitoring.

1. To invoke the installer, right-click the downloaded installer icon and select **Run as Administrator**. The PEM Agent Setup wizard opens, welcoming you.
2. Read and accept the license agreement and then select **Next**.
3. The setup wizard automatically detects an existing agent and upgrades the installed version. Select **Next**.

4. The pemAgent Service Account dialog box might prompt you for the password of the account under which the PEM agent service runs. If prompted, provide the password, and select **Next**.
5. When the Ready to Install dialog box informs you that the installation is about to begin, select **Next**. The wizard upgrades your PEM agent to the latest version.
6. The PEM Agent Setup wizard informs you when the installation completes. Select **Finish**.
7. After the installation completes, you're prompted to restart the machine. Select **Yes** to restart the machine and the PEM agent.

Upgrading the PEM server on a Windows host

The PEM server installer enables you to upgrade between major versions of the PEM server.

1. To invoke the installer, right-click the downloaded installer, and select **Run as Administrator**.
2. The PEM Server Setup wizard welcomes you. Select **Next**.
3. The PEM Server Setup wizard prompts you to accept the license agreement. After reviewing the license agreement, select **I accept the agreement** and select **Next**.
4. The wizard checks the PEM server host for an existing PEM server installation. If the wizard locates an installation, it performs an upgrade. Select **Next**.

Before upgrading the PEM server, the wizard confirms that the requirements of the new PEM server are present. If any supporting components are missing or are at a version that doesn't support the new PEM installation, the wizard informs you that it must upgrade the dependencies. It then invokes the required installers.

1. When the installation wizards completes the dependency upgrades, you're prompted to restart the machine. Select **No** to continue the upgrade process.
2. The wizard then opens the Database Server Installation Details dialog box, prompting you for connection credentials for the database superuser of the PEM backend database. Provide:

- The name of the database superuser in the **User** field.
- The password associated with the database superuser in the **Password** field.

Select **Next**.

1. The pemAgent Service Account dialog box might prompt you for the password of the account under which the PEM agent service runs. If prompted, provide the password, and select **Next**.
2. The Ready to Install dialog box informs you that the setup wizard is ready to perform the installation. Select **Next** to start the installation.

After upgrading the PEM server (and the agent that resides on the same host as the PEM server) and configuring the web service, the PEM setup wizard notifies you of the port on which the service is listening. Use this port number when connecting to the PEM server with the PEM client.

1. Select **OK**. The PEM server setup wizard informs you that the installation is complete.
2. If you're prompted to restart the machine, select **Yes** to restart the machine and the httpd service.

If you installed the PEM backend database server and PEM-HTTPD on different hosts, then you must run the PEM server installer twice: once on each host. Extract the language pack installer, and install it on the host of PEM-HTTPD before invoking the PEM installer. Include the following keywords when invoking the installer to extract the language pack:

```
--extract-languagepack <path>
```

Where `<path>` specifies an existing path for extracting the language pack installer.

> **Note**
>
> By default EDB Language Pack is installed in *C:edblanguagepackv1* .

If you're upgrading the PEM Server using StackBuilder Plus, then you might see an error. After displaying the error, PEM reports that installation is completed. However, the installation isn't complete. You need to do the installation by invoking the installer file from the location where it is downloaded.

After upgrading the PEM server, you might want to upgrade the backend database to a more recent version. For information, see *Upgrading the PEM backend Postgres database* .

11.1.3 Upgrading an HA PEM installation

If you're using a *high availability PEM* , you can perform a major or minor upgrade of the PEM installation.

The example commands provided assume a RHEL-based operating system.

!!!Important This upgrade procedure is designed to minimize downtime to your PEM cluster. However, it restarts your primary database instance once. To allow the primary to restart and avoid one of the replicas being promoted, ensure that automatic failover is disabled.

If you are using Patroni, ensure that the value of `DB_UNIT_FILE` in `/usr/edb/pem/share/.install-config` is set to `patroni` . as detailed in the *installation instructions* .

Upgrade the primary

On the primary server:

1. Update the PEM packages:

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

1. Execute the upgrade script:

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

Upgrade the replicas

On each replica server:

1. Update the PEM packages:

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

1. If you're updating to PEM 10.x from PEM 9 or PEM 10.0, ensure that libpq5 is installed (this is done on the primary by `configure-pem-server.sh`):

```
dnf install libpq5
```

1. Copy the following configuration files from the primary.

```
/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. If you're using NGINX as your web server, also copy the following files from the primary.

```
/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
```

Then reload NGINX and uWSGI:

```
systemctl daemon-reload
systemctl enable edb-uwsgi
systemctl start edb-uwsgi
systemctl restart nginx
```

1. If you're using Apache HTTPD as your web server, also copy the following files from the primary.

```
/etc/httpd/conf.d/edb-pem.conf
/etc/httpd/conf.d/edb-ssl-pem.conf
```

Restart Apache HTTPD:

```
systemctl restart httpd
```

The upgrade is complete.

Upgrading a deployment with separate PEM web application and backend database servers

Upgrade the primary backend

On the primary backend database server:

1. Update the PEM packages:

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

1. Execute the upgrade script:

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

Upgrade the replica backends

On each replica database server:

1. Update the PEM packages:

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

1. Copy the following configuration files from the primary.

```
/usr/edb/pem/share/.install-config
```

Upgrade the web applications

On each web application server:

1. Update the PEM packages:

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

1. Execute the upgrade script:

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

The upgrade is complete.

11.2 Upgrading the PEM backend Postgres database

If you're updating PEM components and the PEM backend database, perform PEM component updates on the server and agent before updating the backend database. For more information about updating PEM component software, see *Upgrading a PEM installation* .

Note

- If your backend database server is earlier than version 13, first upgrade your backend database server and then upgrade the PEM components.

```
-   After upgrading the backend database server, if you encounter this error while␣
↪creating the server in the PEM web interface:
```

```
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.
```

Resolve the error by updating the roles and granting appropriate permissions:

```
UPDATE pem.roles SET rolid = pr.oid FROM pg_roles pr WHERE pr.rolname = pem_ ||␣
↪component;
```

The update process uses the pg_upgrade utility to migrate from one version of the backend server to a more recent version. pg_upgrade enables migration between any supported version of Postgres and any subsequent release of Postgres that's supported on the same platform.

pg_upgrade supports a transfer of data between servers of the same type. For example, you can use pg_upgrade to move data from a PostgreSQL 13 backend database to a PostgreSQL 14 backend database but not to an EDB Postgres Advanced Server 14 backend database. If you want to migrate to a different type of backend database (such as from a PostgreSQL server to EDB Postgres Advanced Server), see *Moving the PEM server* .

You can find more information about using pg_upgrade at pg_upgrade .

1. Download and invoke the updated installer. Installers for PostgreSQL and EDB Postgres Advanced Server are available through the EDB website .

After downloading the installer for the server version you are upgrading to, invoke the installer on the host of the PEM server. Follow the onscreen instructions of the installation wizard to configure and install the Postgres server.

You can optionally use a custom-built PostgreSQL server as a host of the PEM backend database. If you're upgrading from a PostgreSQL backend database listening on port 5432, the new server must be configured to listen on a different port.

2. Configure SSL utilities on the new server. The new backend database must be running the same version of sslutils that the current backend database is running.

The process of configuring sslutils is platform specific. Please refer to the installation documentation for your platform for details of how to install sslutils.

3. Stop the services of both the old backend database and the new backend database.

On RHEL 8.x, open a command line and assume the identity of a superuser. Enter the command:

```
systemctl <service_name> stop
```

Where `<service_name>` specifies the name of the Postgres service.

On Windows, you can use the Services dialog box to control the service. To stop the service: 1. On the Control Panel select **System and Security > Administrative Tools**. 1. Double-click **Services**. 1. In the Services dialog box, select the service name and selet **Stop**.

4. Use the pg_upgrade utility to perform an in-place transfer of existing data between the old backend database and the new backend database. If your server is configured to enforce md5 authentication, you might need to add an entry to the `.pgpass` file that specifies the connection properties (and password) for the database superuser. Or you might need to modify the `pg_hba.conf` file to allow trust connections before invoking `pg_upgrade` . For more information about creating an entry in the `.pgpass` file, see the PostgreSQL core documentation .

During the upgrade process, pg_upgrade writes a series of log files. The cluster owner must invoke pg_upgrade from a directory in which they have write privileges. If the upgrade completes successfully, pg_upgrade removes the log files when the upgrade completes. If you don't want pg_upgrade to delete the upgrade log files, include the `--retain` keyword when invoking pg_upgrade.

To invoke pg_upgrade, assume the identity of the cluster owner, navigate to a directory in which the cluster owner has write privileges, and execute the command:

```
<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>
```

Where:

```
-   `path_to_pg_upgrade`  specifies the location of the pg_upgrade utility. By␣
↪default, pg_upgrade is installed in the `bin`  directory under your Postgres␣
↪directory. -   `old_data_dir_path`  specifies the complete path to the data␣
↪directory of the old backend database. -   `new_data_dir_path`  specifies the␣
↪complete path to the data directory of the new backend database. -   `old_bin_dir_
↪path`  specifies the complete path to the bin directory of the old backend database.
↪ -   `new_bin_dir_path`  specifies the complete path to the bin directory of the␣
↪old backend database. -   `old_port`  specifies the port on which the old server is␣
```

(continues on next page)

(continued from previous page)

```
↪listening. -   `new_port`  specifies the port on which the new server is listening.␣
↪-   `user_name`  specifies the name of the cluster owner.
```

For example, the following command instructs pg_upgrade to migrate the PEM database from PostgreSQL 9.6 to PostgreSQL 11 on a Windows system (if the backend databases are installed in their default locations):

```
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
```

Once invoked, pg_upgrade performs consistency checks before moving the data to the new backend database. When the upgrade is finished, pg_upgrade notifies you that the upgrade is complete.

For detailed information about using pg_upgrade options or troubleshooting the upgrade process, see pg_upgrade .

5. Copy the following certificate files from the `data` directory of the old backend database to the `data` directory of the new backend database:

`ca_certificate.crt`

`ca_key.key`

`root.crt`

`root.crl`

`server.key`

`server.crt`

Once in place on the target server, make sure the files have these platform-specific permissions:

Permissions and ownership on Linux

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--

On Linux, the certificate files must be owned by postgres. You can use the following command at the command line to modify the ownership of the files:

```
chown postgres <file_name>
```

Where file_name specifies the name of the certificate file.

Only the owner of the server.crt file can modify it, but any user can read it. You can use the following command to set the file permissions for the server.crt file:

```
chmod 644 server.crt
```

Only the owner of the other certificate files can modify or read the file. You can use the following command to set the file permissions:

```
chmod 600 <file_name>
```

Where file_name specifies the name of the file.

Permissions and ownership on Windows

On Windows, the service account that performed the PEM server and backend database installation on the target host must own the certificate files moved from the source host. If you invoked the PEM server and Postgres installer using **Run as Administrator** from the context menu of the installer, the owner of the certificate files is Administrators.

To review and modify file permissions on Windows, right-click the file name and select **Properties**.

On the **Security** tab select a group or user name to view the assigned permissions. Select **Edit** or **Advanced** to open dialog boxes that allow you to modify the permissions associated with the selected user.

6. The postgresql.conf file contains parameter settings that specify server behavior. Modify the postgresql.conf file on the new server to match the configuration specified in the postgresql.conf file of the old server.

Use your choice of editor to update the postgresql.conf file of the new server. Modify the following parameters:

- The port parameter to listen on the port monitored by your original backend database (typically set to 5432).
- The ssl parameter to be set to on

You must also ensure that the following parameters are enabled. If the parameters are commented out, remove the pound sign from in front of each postgresql.conf file entry:

- 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'

Your installation might have other parameter settings that require modification to ensure that the new backend database behaves like the old backend database. Review the postgresql.conf files carefully to ensure that the configuration of the new server matches the configuration of the old server.

7. The pg_hba.conf file contains parameter settings that specify how the server enforces host-based authentication. When you install the PEM server, the installer modifies the pg_hba.conf file, adding entries to the top of the file:

11.3 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
```

11.4 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
```

Using your editor of choice, copy the entries from the `pg_hba.conf` file of the old server to the `pg_hba.conf` file for the new server.

8. Restart the service of the new backend database.

On RHEL 8.x, at the command line as superuser enter:

```
systemctl restart <service_name>
```

Where `service_name` is the name of the backend database server.

If you're using Windows, you can use the **Services** dialog box to control the service: 1. In the Control Panel, select **System and Security > Administrative Tools**. 1. Double-click the **Services** icon. 1. In the Services dialog box, select the service name and start the service.

11.5 Upgrading SQL Profiler

If you are upgrading from v4.0 (or an earlier version) of SQL Profiler, where the functions where created by loading the SQL file included in a plugin, first migrate to the extension and later update to 4.1:

```
#  running as superuser

CREATE EXTENSION sql_profiler VERSION 4.0;

ALTER EXTENSION sql_profiler UPDATE TO 4.1;
```

11.6 Upgrading the PEM web server

The PEM web server is used to serve the PEM web application. On Linux, the web server can be NGINX or Apache HTTPD, the default for new installations from PEM 10.0 onwards is NGINX. PEM doesn't bundle the web server but instead uses the packages provided by the system package manager for your chosen web server. You can *Upgrading the web server on Linux* as required using your package manager.

On Windows, the PEM installer bundles a version of Apache HTTPD called PEM-HTTPD. Therefore you can only *update the web server* using an EDB installer.

11.6.1 Upgrading the web server on Linux

Important

If you're using Red Hat Enterprise Linux (RHEL), Rocky Linux, AlmaLinux, or Oracle Linux, and you're updating the web server to address a vulnerability, read *Versioning in RHEL and RHEL derivatives* .

If you're running Linux, manage the webserver along with other system software on your servers. We recommend always using the latest version available from your package manager.

You don't need to stop PEM or back up any files prior to upgrade. Initiate the upgrade from your package manager.

RHEL / Rocky Linux / AlmaLinux / Oracle Linux

Run the following command, substituting `<web-server>` for `nginx` or `httpd` as appropriate.

```
sudo yum update <web-server>
```

Debian / Ubuntu

Run the following command, substituting `<web-server>` for `nginx` or `apache2` as appropriate.

```
sudo apt update
sudo apt upgrade <web-server>
```

SLES

Run the following command, substituting `<web-server>` for `nginx` or `apache2` as appropriate.

```
sudo zypper update <web-server>
```

Versioning in RHEL and RHEL derivatives

You might notice that the latest version of NGINX or HTTPD available from RHEL or other RHEL derivatives has a significantly older version number than the latest community release. Specifically, RHEL 8 provides NGINX 1.22 and HTTPD 2.4.37. RHEL 9 provides NGINX 1.22 and HTTPD 2.4.62. It's very important to understand that Red Hat builds custom NGINX and HTTPD packages for RHEL and backports security fixes.

For this reason, don't assume that a vulnerability present in the community release is present in the RHEL package of the same version. If you're trying to install a newer version of HTTPD for this reason, it's almost certainly not necessary. See the NGINX versions supported by Red Hat and Apache HTTPD versions supported by Red Hat articles in the Red Hat knowledge base for more information.

> **Warning**
>
> Red Hat doesn't support NGINX or HTTPD packages from the community or built from source. If you're considering using any other source of NGINX or HTTPD on a RHEL system, read the Apache HTTPD versions supported by Red Hat article and make sure you understand the implications.

11.6.2 Upgrading PEM-HTTPD on Windows

PEM-HTTPD is bundled with the PEM Windows installer and is also available through Stack Builder. In general it is recommended that you update your PEM installation regularly. Your PEM-HTTPD installation will be automatically updated when you update the PEM application.

If you do not wish to update PEM, or to apply a PEM-HTTPD release which is not yet available in a PEM installer, you can use StackBuilder.

To upgrade using StackBuilder download the latest PostgreSQL or EDB Advanced Server installer for Windows and use Stack Builder

or StackBuilder Plus respectively to update PEM-HTTPD. PEM-HTTPD is located under the Web Development category.

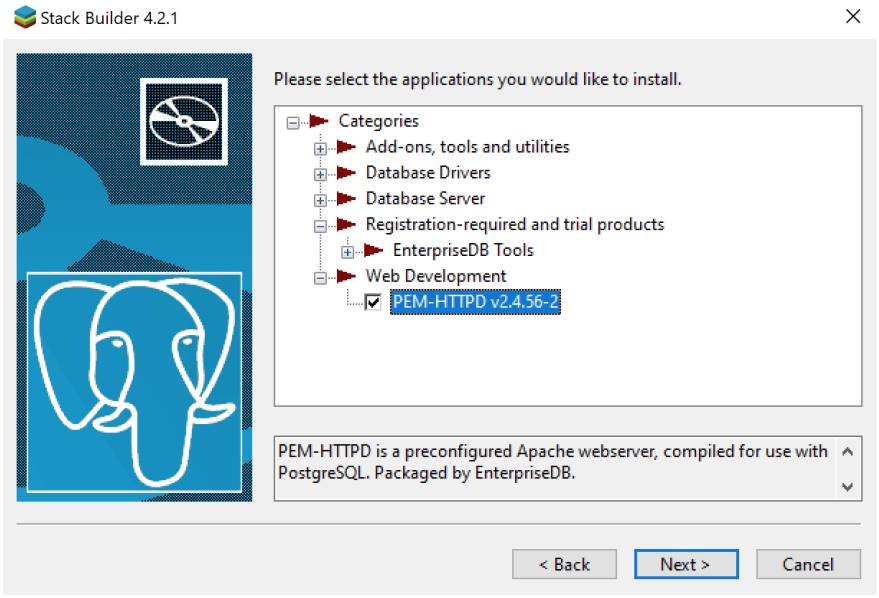


Fig. 1: PEM-HTTPD in Stack Builder

11.7 Moving the PEM server

You can move a PEM server from one host machine to a new host machine. The PEM server on the new host (the target) must be installed with the same version of the PEM server installer as the original host (the source). If you don't use the same installer version, you might encounter a schema-mismatch error.

The backend database of the target server (either PostgreSQL or EDB Postgres Advanced Server) can have the same type and version or a different type and version from the backend database of the source PEM server. You can migrate a PEM server that resides on a PostgreSQL host to an EDB Postgres Advanced Server host and vice versa.

Before starting the server migration, make sure that the firewalls between the source host, the target host, and the host of any PEM agent allows connections between the services.

1. Prepare the target host.

Invoke the installer for the PEM server on the target host. You must use the same version of the PEM server installer that you used when installing the source PEM server.

The backend database of the target server can have a different version or type from the backend database of the source. If the new PEM server doesn't reside on the same type of backend database as the original server, you must ensure that the same version of the sslutils extension is installed on the new server host. The version of sslutils that's distributed with the PEM installers is freely available for download from the EDB website .

For information about installing the PEM server or the sslutils extension, see the PEM installation steps .

2. Drop existing schemas from the new PEM server.

The migration process re-creates the `pem` , `pemdata` , and `pemhistory` schemas from the source PEM server on the target PEM server. To prepare for the move, use the psql client to delete these schemas from the `pem` database on the target host. You can open the psql client at the command line or by selecting **Postgres Enterprise Manager > SQL Shell (psql)**.

When the psql client opens, connect to the `pem` backend database as the database superuser. After connecting to the `pem` database on the target host, drop the schemas:

```
DROP SCHEMA pem CASCADE;

DROP SCHEMA pemdata CASCADE;

DROP SCHEMA pemhistory CASCADE;
```

When dropping the schemas, you must include the `CASCADE` keyword, instructing the server to delete all dependent objects. When executing the command, the psql client displays a list of the dependent objects. The client confirms each the schema is removed by displaying `DROP SCHEMA` .

3. Prepare the PEM agents on the new PEM server.

Before moving the PEM server, you must identify the number of agents that are monitored by the source PEM server and create identities for that number of agents, less one, on the target server. To discover the total number of PEM agents monitored by the PEM server, connect to the `pem` database on the source host with the psql client, and query the `pem.agent` table.

```
SELECT id FROM pem.agent WHERE active = true;
```

You must manually create the number of agents that reside on the original PEM server, less one. (The PEM server installer creates one agent on the target host.) For example, if the source server contains three agents, you must manually create two more agents. Open a psql session with the `pem` database on the target server, and create the required agents:

```
CREATE USER agent<X>;
```

Where `<X>` specifies an agent number. `agent1` is created on the target host by the PEM server installer.

Then, use the `GRANT` command to assign each agent that resides on the target PEM server `pem_agent` permissions:

```
GRANT pem_agent TO agent<X>;
```

Where `<X>` specifies an agent number.

4. Generate a backup script of the source PEM server.

You can use the pg_dump utility to generate a script that contains the commands required to re-create the `pem` database on the target host. By default, pg_dump is installed in the `bin` directory under your Postgres installation. To invoke pg_dump, in the `bin` directory, enter:

```
pg_dump -U <user_name> <db_name> > <file_name>
```

Where:

```
-   `<user_name>`  specifies the name of the database superuser for the PEM backend␣
↪database. -   `<db_name>`  specifies the name of the PEM backend database. -   `
↪<file_name>`  specifies the name of the script generated by pg_dump.
```

When prompted, provide the password associated with the user specified.

The command shown instructs pg_dump to generate a script that, when executed, re-creates the `pem` database. The script is named `backup.sql` and is created in the `tmp` directory. pg_dump is connecting to the server using the credentials of the user postgres.

Invoking the pg_dump utility doesn't interrupt current database users.

5. Move the backup to the target host.

Move the script generated by the pg_dump utility to the target host of the PEM server.

6. Restore the backup on the target host.

On the target host, in the `bin` directory under the Postgres backend database installation directory, start psql, executing the script generated by the pg_dump utility:

```
psql -U <user_name> -d pem -f <file_name>
```

Where:

```
- `<user_name>` specifies the name of the database superuser. The user specified␣
↪must have connection privileges for the backend database. - `<file_name>` ␣
↪specifies the complete path to the backup script generated by pg_dump.
```

When prompted, provide the password associated with the database superuser.

The example shown uses the psql client to invoke a script named `backup.sql` to recreate the `pem` database. The script is invoked using the privileges associated with the database superuser postgres.

7. Stop the database server on the target host.

To stop the PEM server on RHEL 8.x, use the command:

```
systemctl stop <service_name>
```

Where `<service_name>` specifies the name of the backend database server. For a PostgreSQL backend database, the service name is `postgresql-<x>` . For an EDB Postgres Advanced Server backend database, the service name is `edb-as-<X>` , where `<X>` specifies the version number.

If you're using Windows, you can use the Services dialog box to control the service. To open the Services dialog box, from the Control Panel, select **System and Security > Administrative Tools**. Double-click the **Services** icon. In the Services dialog box, select the service name in the list, and select **Stop**.

8. Copy the certificate files to the target host.

You must replace the certificate files that are created when the target host is installed with the certificate files of the source host. Copy the following files from the source PEM server to the target PEM server:

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

Copy the files to the `data` directory under the Postgres installation that provides the backend database for the target cluster.

On Linux, the files reside in:

```
/var/lib/pgsql/<X>/data/
```

On Windows, the files reside in:

```
C:\Program Files\PostgreSQL\<X>\data
```

Where:

`<X>` specifies the version of PostgresSQL on your system.

The files already exist on the target cluster. Delete the existing files before performing the copy, or overwrite the existing files with the files from the source server. Once in place on the target server, the files must have the platform-specific permissions shown.

On Linux

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--

On Linux, the certificate files must be owned by postgres. Use the following command to modify the ownership of the files:

```
chown postgres <file_name>
```

Where `file_name` specifies the name of the certificate file.

Only the owner of the `server.crt` file can modify the file, but any user can read it. Use the following command to set the file permissions for the `server.crt` file:

```
chmod 644 server.crt
```

Only the owner of the other certificate files can modify or read the files. Use the following command to set the file permissions:

```
chmod 600 <file_name>
```

Where `file_name` specifies the name of the file.

On Windows

On Windows, the service account that performed the PEM server and backend database installation on the target host must own the certificate files moved from the source host. If you invoked the PEM server and Postgres installer using **Run as Administrator** from the installer context menu, the owner of the certificate files is Administrators.

To review and modify file permissions on Windows, right-click the file name and select **Properties**. On the **Security** tab, select a group or user name to view the assigned permissions. Select **Edit** or **Advanced** to open dialog boxes that allow you to modify the permissions associated with the selected user.

9. Move the PEM agent certificate files to the PEM server host.

You must move the certificate files used by the PEM agent of the source PEM server to the target host. This step is platform specific.

On Linux

Copy the `agent1.key` and `agent1.crt` files from the source host to the target host. By default, on Linux, the files are installed in `/root/.pem` . Copy the files to the same directory on the target host.

File ownership and permissions of the files must be set to:

File name	Owner	Permissions
agent1.key	root	-rw-------
agent1.crt	root	-rw-r--r--

If necessary, navigate to `/root/.pem`, and use the following commands to modify the permissions and ownership of the `agent1.key` file:

```
chmod 600 agent1.key

chown root agent1.key
```

Use the following commands to modify the permissions and ownership of the `agent1.crt` file:

```
chmod 644 agent1.crt

chown root agent1.crt
```

On Windows

Copy the `agent1.key` and `agent1.crt` files from the source host to the target host. On Windows, the files are located in:

```
C:\Users\<user_name>\AppData\Roaming\pem
```

Where `user_name` is the name of the user that invoked the PEM installer.

The ownership and permissions associated with the certificate files on the target machine must match the ownership and permissions of the certificate files on the source machine. If you invoked the PEM server and Postgres installer using **Run as Administrator** on the installer context menu, the owner of the agent certificate files is Administrators.

To review and modify file permissions on Windows, right-click the file name and select **Properties**. On the **Security** tab, select a group or user name to view the assigned permissions. Select **Edit** or **Advanced** to open dialog boxes that allow you to modify the permissions associated with the selected user.

10. Update the `pg_hba.conf` files on the target host.

Modify the `pg_hba.conf` file on the target host to allow connections from each PEM agent. By default, the `pg_hba.conf` file is located in the data directory under your Postgres installation.

11. Start the server on the target host.

After modifying the `pg_hba.conf` file, you must restart the server for the changes to take effect.

To restart the database server on Linux, use the command:

```
/etc/init.d/<service_name> start
```

Where `service_name` is the name of the backend database server.

On Windows, you can use the Services dialog box to control the service. To open the Services dialog box, on the Control Panel, select **System and Security > Administrative Tools**. Double-click the **Services** icon. When the Services dialog box opens, select the service name in the list, and start the service.

12. Connecting monitored agents to the new PEM server host.

To instruct existing PEM agents to connect to the new PEM server host, you must:

- Ensure that the PEM agent host can connect to the new PEM server host.

- Modify the registry on each Windows host with a PEM agent or the agent configuration files on each Linux host with a PEM agent, specifying the IP address and port of the new PEM server.
- Restart the PEM agent's service. These steps are platform specific: - *If the PEM agent resides on Linux* - *If the PEM agent resides on Windows*

11.7.1 If the PEM agent resides on Linux

Use your choice of editor to modify the `agent.cfg` file, specifying the new IP address and port number of the PEM server in the `pem_host` and `pem_port` parameters.

By default, the `agent.cfg` file is located in:

`/usr/edb/pem/agent/etc/agent.cfg`

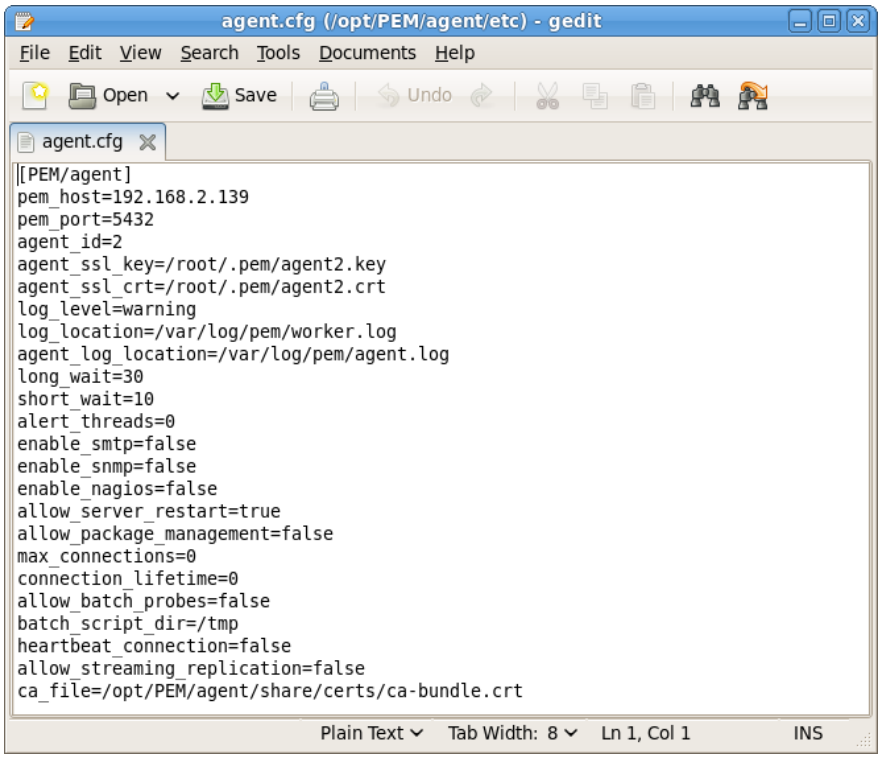


Fig. 2: The agent.cfg file

After modifying the `agent.cfg` file, you must restart the PEM agent service. You can use the `pemagent` service script on the Linux command line to restart the service:

```
/etc/init.d/pemagent restart
```

11.7.2 If the PEM agent resides on Windows

Before modifying the Windows registry on the monitored node, confirm that the firewall on the host of the PEM agent allows connections to the PEM server. After confirming that the PEM agent host can connect to the PEM server host, you can use the Windows Registry Editor to review and edit the `PEM_HOST` and `PEM_PORT` entries to ensure that they correctly identify the host and port used by the PEM server. To open the Registry Editor, enter `regedit` in the Windows Run dialog box or in the Windows start menu search box. Navigate through the registry tree control to view or modify registry entries.

The PEM agent registry entries are located at: `HKEY_LOCAL_MACHINE\Software\EnterpriseDB\PEM\agent`

The PEM_HOST and PEM_PORT entries must specify the address and port number of the new PEM server on the target host. To modify a registry entry, right-click the entry name and select **Modify** from the context menu. Then use the Edit String dialog box to make any changes to the value of the entry. After you finish, select **OK**.

After modifying the registry, you must restart the PEM agent's service. You can use the Services dialog box, accessed through the Windows Control Panel, to restart the Postgres Enterprise Manager - pemagent service.

After moving the server, change the connection properties in any installed PEM clients to connect to the new host of the PEM server, agents, and monitored servers.

Note

After moving the server, if you encounter this error while creating the server in the PEM web interface:

```
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.
```

Resolve the error by updating the roles and granting appropriate permissions:

```
UPDATE pem.roles SET rolid = pr.oid FROM pg_roles pr WHERE pr.rolname = pem_ ||␣
↪component;
```

CHAPTER 12

Splitting an existing PEM server

You can split an existing PEM installation into two installations. Splitting is useful if your original PEM instance has become overloaded and you want to split the load over two servers. It can also be useful if you're restructuring your database estate and want to take a segmented approach to monitoring.

You can split an existing PEM instance into three or more instances by splitting each new instance. You can also set up a new PEM instance and move a portion of the monitored servers and agents to the new instance.

> **Note**
>
> The example commands are for PostgreSQL 16. Modify the service name and file paths as appropriate for your environment.

12.1 Install PEM on the new host

!!!Important PEM and Postgres versions must match Ensure the versions of Postgres and PEM in your new environment match the versions used in the existing environment.

1. Install the PEM server package on the new host .
2. *Configuring the PEM server on Linux* .

12.2 Copy all data from your existing PEM instance

On the new PEM host:

1. Stop the Postgres and PEM agent services:

```
systemctl stop postgresql-16
systemctl stop pemagent
```

2. Delete the Postgres data directory:

```
rm -rf /var/lib/pgsql/16/data/*
```

3. Clone the the existing PEM data in your new instance:

```
su - postgres
pg_basebackup -h <OLD_PEM_IP> -p <OLD_PEM_SERVER_PORT> -U postgres \
-D /var/lib/pgsql/16/data -Fp -Xs -P
```

12.3 Update the pg_hba.conf file on the new PEM instance

At this point, the `pg_hba.conf` file on the new PEM instance is a copy of the one on the existing PEM instance. If any entries reference the external IP of the existing PEM server (for example, `host ... <old_ip> ...`), modify them to use the new PEM server IP address or `localhost` /`127.0.0.1` , as appropriate.

You can now start the Postgres and PEM agent services. (You might need to restart them again later.)

```
systemctl start postgresql-16
systemctl start pemagent
```

12.4 Configure SSL on the new server and reconfigure PEM agents

How you configure SSL depends on how you want to manage SSL certificates. From the four approaches, select the one that meets your needs.

All of these scenarios are viable if you're using unique agent users or a single common agent user.

!!!Note External certificate authorities If you exclusively use SSL certificates provided or signed by an external certificate authority, go to *Approach 4 – Use only certificates from an external CA* .

!!!Note Web server certificates These instructions are for configuring SSL for connections to Postgres. For the PEM web application, generate a new self-signed certificate during server configuration. You can replace the web server certificate by following the documentation .

12.4.1 Approach 1 – Reuse certificates from the existing PEM instance

Use this approach if you want to use the same server certificates for both PEM instances. This approach requires no changes to any SSL settings or files.

> **Note**
>
> In general, we recommend Approach 2 over Approach 1 because it ensures each PEM server has a unique identity. This is good practice and is required if you want to use stricter client SSL modes such as *verify-full* .

1. Copy the PEM agent certificates from the old PEM instance to the new PEM instance. (By default, these are located in `/root/.pem` on Linux and `C:\Users\\AppData\Roaming\pem` on Windows.)
2. Restart Postgres and the PEM agent services on the new system.
3. For all agents you want to migrate to the new PEM instance, update the `pemhost` parameter in the `agent.cfg` file of all agents to the address of the new PEM instance.
4. Restart the affected PEM agents.

12.4.2 Approach 2 – Use different server certificates but the same CA

Use this approach if you want to have different server certificates on the two PEM servers and want to allow both PEM servers to act as a single certificate authority.

Using this approach, agent certificates issued by either PEM server will be signed identically and trusted by both servers. This is convenient as it avoids the need to replace any agent certificates while still ensuring the servers have distinct identities.

1. Back up the existing `server.crt` and `server.key` located in the Postgres data directory.
2. Generate a new `server.key` and CSR for the new PEM server.

```
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. Sign the CSR, and generate a server certificate with your existing CA certificate.

```
openssl x509 -req -days 365 -in server.csr -CA ca_certificate.crt -CAkey ca_key.key -
↪CAcreateserial -out server.crt
```

4. Change the owner and the permissions on the generated `server.crt` file.

```
chown postgres /var/lib/pgsql/16/data/server.crt
chmod 600 /var/lib/pgsql/16/data/server.crt
```

5. Restart Postgres and PEM agent services on the new system.
6. For all agents you want to migrate to the new PEM instance, update the `pemhost` parameter in the `agent.cfg` file of all agents to the address of the new PEM instance.
7. Restart the affected PEM agents.

12.4.3 Approach 3 – Replace all server and agent certificates

Use this approach if you want to have completely distinct server certificates and certificate authorities for the two servers. Using this approach, a given agent can authenticate only with its "parent" PEM server. The two PEM instances will have no SSL configuration in common.

!!!Important This approach requires generating new client certificates for every agent. Unless you have orchestration tooling capable of doing this or are using a *common agent user* , this can be a significant effort.

1. Follow the instructions for *Regenerating the server SSL certificates* to regenerate the following files:
   - `server.crt`
   - `server.key`
   - `root.crt`
   - `root.crl`
   - `ca_certificate.crt`
   - `ca_key.key`
5. Restart Postgres and PEM agent services on the new system.
6. For all agents you want to migrate to the new PEM instance, update the `pemhost` parameter in the `agent.cfg` file of all agents to the address of the new PEM instance.

7. *Regenerating the agent SSL certificates* for each agent user and copy the new certificates and keys into place. You must also regenerate the certificate and key and for the agent on the PEM server.
8. Restart the affected PEM agents.

12.4.4 Approach 4 – Use only certificates from an external CA

Use this approach if you exclusively use SSL certificates provided or signed by an external certificate authority.

Because your existing agent certificates were issued by a trusted authority that's included in the Postgres server CA cert, both PEM servers can validate all your agent certificates, meaning all agents are already able to connect to either PEM server. You need to request a new server certificate only for the new PEM server.

1. Obtain a new server certificate and key from your provider, supplying the details of the new PEM instance.
2. Replace `server.crt` and `server.key` (or whatever files are currently specified by `ssl_server_cert` and `ssl_server_key`) with your new certificate and key respectively.
3. Restart Postgres.

12.5 Disable disconnected agents on each PEM instance

You now have two PEM instances, with your existing agents split between them. However, both instances contain records for all agents. Clean up the PEM instances to reduce wasted resources and false alerts by marking the agents that aren't connected to each instance as inactive.

In the PEM web application, delete the agent and its monitored servers. In Object Explorer, right-click each one and select **Remove**.

Alternatively, you can perform this action through SQL. Execute the following on the `pem` database:

```
UPDATE pem.agent SET active = false WHERE id in (list_of_ids);
UPDATE pem.server SET active = false WHERE id in (list_of_ids);
```

12.6 Validate your PEM instances

Open the PEM monitoring dashboard of the new instance.

- Confirm that only the required servers and agents are visible.
- Verify that these servers/agents show an active status and are sending metrics.
- Check the agent logs for any connection or authentication errors.

CHAPTER 13

Uninstalling

To uninstall PEM components see:

- *Uninstalling Postgres Enterprise Manager components on Linux*
- *Uninstalling Postgres Enterprise Manager components on Windows*

13.1 Uninstalling Postgres Enterprise Manager components on Linux

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

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

Note

Before uninstalling the PEM Agent, you need to de-register the agent first. You can de-register the agent using the *pemworker* command-line utility. After that, you can proceed with the uninstallation steps.

13.1.1 Uninstalling PEM components from RHEL or Rocky Linux or AlmaLinux hosts

You can use variations of the `rpm` command to remove the installed packages from Rocky Linux/AlmaLinux/RHEL 8.x hosts. Also, you can use the `dnf remove` command to remove the installed package from Rocky Linux/AlmaLinux/RHEL 8.x hosts. Note that removing a package does not damage the PEM data directory.

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

```
rpm -e package_name
```

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

- You can use `dnf remove` command to remove the pem server or Agent along with the `edb-pem` and `edb-pem-docs` dependencies on Rocky Linux or AlmaLinux or RHEL 8.x hosts. To remove a package, open a terminal window, assume superuser privileges, and enter the command:

```
dnf remove package_name
```

13.1.2 Uninstalling PEM components from Debian or Ubuntu hosts

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

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

```
apt-get remove package_name
```

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

```
apt-get purge package_name
```

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

13.1.3 Uninstalling PEM components from SLES hosts

To uninstall PEM server or Agent from a SLES host, invoke the following command:

```
zypper remove package_name
```

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

13.2 Uninstalling Postgres Enterprise Manager components on Windows

If you uninstall the PEM server from a host, the PEM agent installed on the same host is uninstalled. But if you uninstall the PEM agent, then the PEM server installed on the same host will not be uninstalled.

You can use the Windows `Add/Remove Programs` application to remove PEM components from a Windows host. Select the `Add/Remove Programs` option from the Windows `Control Panel` . When the `control panel` opens, locate the name of the PEM component in the program list. Click the `Remove` button to remove the component.

You can also invoke the uninstaller that resides at the following location:

For the PEM Server, `C:\Program Files\edb\pem\server\uninstall-pemserver`

For the PEM Agent, `C:\Program Files\edb\pem\agent\uninstall-pemagent`

CHAPTER 14

Troubleshooting

14.1 Known installation and configuration issues

14.1.1 Conflict with existing mod_wsgi

If you have a conflicting version of mod_wsgi on your system, the following error appears while installing the PEM server on RHEL, Rocky, Alma or Oracle Linux:

```
[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]#
```

To resolve the issue, remove the `python3-mod_wsgi` package and try installing PEM again.

```
dnf remove python3-mod_wsgi
```

14.1.2 Errors from JIT

If JIT is enabled on the PEM backend Postgres instances, you may see this error in the worker.log after configuring the PEM server on RHEL, Rocky, Alma or Oracle Linux.

```
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
```

To resolve the error, set the `jit` parameter to `off` in `postgresql.conf` file of the backend database server:

```
jit=off
```

Restart the backend database server.

14.1.3 Crash due to bug in RHEL Python 3.9 standard library

If your RHEL 9, Rocky 9, Alma 9, or Oracle Linux 9 server is running an impacted Python 3.9 version, an error appears during PEM configuration that ends with:

```
TypeError: realpath() got an unexpected keyword argument strict
```

You can resolve the error by updating to the latest Python 3.9 packages. We have verified the issue isn't present in the latest release of Python 3.9.21 as of October 2025. However, it was present in some earlier releases of 3.9.21 as well as earlier Python versions.

```
dnf upgrade python3
```

Retry PEM configuration.

14.1.4 Blocking TRACK/TRACE Methods in Apache HTTPD

!!!Note Version and configuration compatibility This configuration is pre-configured by default in the following versions of PEM:

- PEM 9 _ Version 9.8.1 and later
- PEM 10 _ Version 10.3.0 and later

In versions prior to those listed above, the configuration string exists within the file but is positioned incorrectly. It must be manually moved into the specific `VirtualHost` block governing the PEM HTTPS endpoint to take effect.

```
<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>
```

- For Apache on Ubuntu, Debian, or SLES15, update the `/etc/apache2/sites-enabled/edb-ssl-pem.conf` file.
- For HTTPD on RHEL, update the `/etc/httpd/conf.d/edb-ssl-pem.conf` file.

14.2 Reconfiguring the PEM server

In some situations, you might need to uninstall the PEM server, reinstall it, and then configure the server again. To do so:

1. Remove the PEM server configuration and uninstall:

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

2. Remove the PEM packages:

```
yum erase edb-pem-server
```

3. Drop the `pem` database:

```
DROP DATABASE pem
```

4. Move the certificates from `/root/.pem/` to another location:

```
mv /root/.pem/* <new_location>
```

5. Move the `agent.cfg` file from `/usr/edb/pem/agent/etc/agent.cfg` to another location:

```
mv /usr/edb/pem/agent/etc/agent.cfg <new_location>
```

6. Move the configuration data directory to another location:

```
mv /var/lib/pemhome/.pem <new_location>
```

7. Then, configure the PEM server again:

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

14.3 PEM web application not loading

If the PEM web application isn't loading, *check the web services are running* . If they are running, check the web server and application error logs.

If you are using NGINX as the web server, the web server logs are located at `/var/log/nginx/error.log` and the application logs are at `/var/log/edb-uwsgi/pem.log`

If you are using Apache HTTPD The log is located at `/var/log/httpd/error_log` . This file contains both web server and application error messages.

If you are using Apache HTTPD as the web server, and see the following message in the log:

```
Truncated or oversized response headers received from daemon process edbpem: /usr/edb/
↪pem/web/pem.wsgi
```

Add the following statement at the bottom of the Apache `httpd.conf` file located in the `/etc/httpd/conf` folder:

```
WSGIApplicationGroup %{GLOBAL}
```

Restart the HTTPD server after adding the statement:

```
sudo systemctl restart httpd.service
```

14.4 Error connecting to PostgreSQL server

When connecting to a PostgreSQL server, you might get one of these error messages. Review the message carefully. Each error message attempts to incorporate the information you need to resolve the problem.

- **Connection to the server has been lost**: This error message indicates that the connection attempt took longer than the specified threshold. There might be a problem with the connection properties provided on the Server dialog box, network connectivity issues, or the server might not be running.
- **Could not connect to Server: Connection refused**: There are two possible reasons for this error:
  - The database server isn't running. Start the server.
  - The server isn't configured to accept TCP/IP requests on the address shown.

For security reasons, a PostgreSQL server "out of the box" doesn't listen on TCP/IP ports. Instead, you must enable it to listen for TCP/IP requests. Add `tcpip = true` to the `postgresql.conf` file for Versions 7.3.x and 7.4.x. Add `listen_addresses='*'` for Version 8.0.x and above. These additions make the server accept connections on any IP interface.

For more information, refer to the PostgreSQL documentation about runtime configuration .

- **FATAL: no pg_hba.conf entry**: If PEM displays this message when connecting, your server can be contacted correctly over the network, but it isn't configured to accept your connection. Your client wasn't detected as a legal user for the database.

To connect to a server, configure the `pg_hba.conf` file on the database server to accept connections from the host of the PEM client. Modify the `pg_hba.conf` file on the database server host, and add an entry in the form:

- `host template1 postgres 192.168.0.0/24 md5` for an IPV4 network
- `host template1 postgres ::ffff:192.168.0.0/120 md5` for an IPV6 network

For more information, see the PostgreSQL documentation about client authentication .

- **FATAL: password authentication failed**: The `password authentication failed for user` error message indicates there might be a problem with the password you entered. Retry the password to confirm you entered it correctly. If the error message returns, make sure that you have the correct password, that you are authorized to access the server, and that the access was correctly configured in the server's `postgresql.conf` configuration file.

14.5 PEM web server connectivity check

Run this command to check whether your client can reach the PEM server and obtain a response:

```
curl https://<SERVER_ADDR>:8443/pem/misc/ping -k -i
__OUTPUT__
PING
```

Where `<SERVER_ADDR>` is the IP address of your PEM server. The output `PING` confirms the PEM web server is up and running.

14.6 SQL Error - column "ssl" of relation "server" does not exist

An incompatibility between specific PEM agent and PEM server versions can cause server registration via pemworker to fail with a "missing column" SQL error.

Affected Scenarios:

- PEM agent 10.3 paired with any PEM server earlier than 10.3.
- PEM agent 10.1.1 paired with PEM server 10.1.1 (Note _ version 10.1.2 is unaffected).

This issue is most frequent during staged upgrades where agents are updated before the central server.

Recommended Resolutions:

- Complete the upgrade _ Ensure both the PEM server and agent are updated to matching versions (10.3 or later) before registering new servers.
- Use the Web UI _ Register the server through the PEM web interface instead of the command line.
- Temporary downgrade _ Revert the PEM agent to an older version to perform the registration, then proceed with the agent upgrade.

14.7 SSL Error - Incomplete sslutils upgrade

If the sslutils system package is upgraded to version 1.4 but the corresponding database extension remains on an earlier version, agent registration will fail with the following OpenSSL error _ `Error_setting_validity_before_time` .

- **Resolution** _ To synchronize the database extension with the system package, run the following command on the PEM backend database:

```
ALTER EXTENSION sslutils UPDATE TO 1.4;
```

14.8 Nginx SSL_read() failure errors

If you see the following error in the Nginx error log, it indicates that a client closed the connection before the server could send a response. This is typically a client-side issue and not a problem with the PEM server itself.

```
SSL_read() failed (SSL: error:0A000126:SSL routines::unexpected eof while reading)␣
↪while keepalive
```

Cause

This issue occurs when a client initiates a request to the PEM server but then closes the TLS connection without sending a `close_notify` alert. The Nginx error log captures this as an SSL read failure due to an unexpected end of file (EOF) while reading.

It affects environments where:

- The OpenSSL version is 1.1.1e or 3.0.x
- The Nginx version is earlier than 1.21.2

Resolution

Upgrade Nginx to version 1.21.2 or later, which includes a fix for this issue. If you are using the Nginx package provided by your Linux distribution, check for updates that include this version.

This example shows how to install a compatible Nginx version (1.24.0) on an EL9-based system and restart the service:

```
#  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
```

After upgrading and restarting Nginx, the `SSL_read()` failure errors should no longer occur when clients close connections prematurely.

CHAPTER 15

Changing the default port

By default, the 8443 port is assigned for the web services when the PEM server is configured. You can change the port after configuration by changing a few parameters in the web server configuration files. The names and locations of these files are platform specific.

15.1 For RHEL

15.1.1 Modify the SELinux configuration

By default, SELinux is enabled in RHEL. Ensure that access to the configured port number is permitted.

1. If `semanage` isn't installed, install it:

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

2. Check whether your port is listed:

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

3. If your port doesn't appear in the list, run this command:

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

15.1.2 Configure the web server (NGINX)

1. Edit the file `/etc/nginx/conf.d/edb-pem.conf` , replacing 8443 with your port number in the following section:

```
server {
   listen  80;
   listen  [::]:80;
   server_name  HTTPD-EDBPEM-SERVER-v10;
```

(continues on next page)

(continued from previous page)

```
   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. Restart the nginx service:

```
sudo systemctl restart nginx
```

15.1.3 Configure the web server (Apache HTTPD)

1. Edit the file `/etc/httpd/conf.d/edb-ssl-pem.conf` , replacing 8443 with your port number in the following parameters:

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

1. Restart the httpd service:

```
sudo systemctl restart httpd
```

You can now access the PEM web interface using your port. For more details, see *Accessing the web interface* .

15.2 For Debian and Ubuntu

15.2.1 Configure the web server (NGINX)

1. Edit the file `/etc/nginx/sites-available/edb-pem.conf` , replacing 8443 with your port number in the following section:

```
server {
   listen  80;
```

(continues on next page)

(continued from previous page)

```
   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. Restart the nginx service:

```
sudo systemctl restart apache2
```

15.2.2 Configure the web server (Apache HTTPD)

1. Edit the file `/etc/apache2/sites-available/edb-ssl-pem.conf`, replacing 8443 with your port number in the following parameters:

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

1. Restart the httpd service:

```
sudo systemctl restart apache2
```

You can now access the PEM web interface using your port. For more details, see *Accessing the web interface* .

CHAPTER 16

Registering a PEM agent

Before you can use a PEM agent, you must register it with a PEM server. **PEM agents installed by the PEM server package are registered automatically during server configuration.** For all other agents, follow these instructions.

> **Note**
>
> After upgrading the PEM agent, you need to restart it. You don't need to register it again.

16.1 How to register PEM agents

On Linux and Windows hosts, the PEM agent package includes a command line utility called pemworker. You can use it to perform management tasks, including *Registering a PEM agent using the pemworker utility* .

On Windows, the PEM agent graphical installer allows you to register the agent when installing it. This convenience option doesn't support all the possibilities provided by the pemworker utility. If you don't want the installer to register the agent, clear the **Register now** checkbox. For more details, see the installation instructions .

16.2 Registering a PEM agent using the pemworker utility

The pemworker utility is installed with the PEM agent. It's located in `/usr/edb/pem/agent/bin` on Linux and `C:\Program Files\edb\pem\agent-x64\bin` on Windows.

To register an agent, set the PEM server password, invoke the utility as shown in the examples, and add the relevant options from the table as needed. Follow each option with a corresponding value.

16.2.1 Linux

```
export PEM_SERVER_PASSWORD=edb

#  Running as root
```

(continues on next page)

(continued from previous page)

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

16.2.2 Windows

```
set PEM_SERVER_PASSWORD=edb

#  Running as admin

./pemworker.exe REGISTER
```

Option	Description
--pem-server	The IP address of the PEM backend database server. This parameter is required unless *--pem-registration-conn-string* is specified.
--pem-port	The port of the PEM backend database server. The default value is *5432*.
--pem-user	The name of a database user with the pem_admin role and the *rolcreaterole* flag set on the PEM backend database server (the latter is not required if you specify an existing user via *--pem-agent-user* or an agent connection string), or a superuser. This user will be used to connect to the PEM server to perform agent registration. This parameter is required unless *--pem-registration-conn-string* is specified.
--pem-agent-user	The name of a database user on the PEM backend database server. After registration, the agent will use this user to open connections to the PEM database server to write probe data, evaluate alerts, and so on. This parameter is optional. If omitted, the agent connects using a new user named *agent<N>* created during registration, where *<N>* is the agent ID.
--pem-ssl-mode	The *Deciding on an SSL model* for the PEM agent user to use (see above). The possible values are *prefer*, *require*, *disable*, *verify-CA*, and *verify-full*. The default value is *require*.
--pem-registration-conn-str	The connection string used to connect to the PEM server during registration. See *Using connection strings* .
--cert-path	The complete path to a directory in which to store certificates. If you don't provide a path, certificates are created in *~/.pem* on Linux and *%APPDATA%/pem* on Windows.
--config-dir	The directory path for the configuration file. The default is *<pemworker path>/../etc*.
--display-name	A user-friendly name for the agent to display in the PEM browser tree. The default is the host's fully qualified domain name (FQDN), falling back to the hostname if this option isn't set.
--force-registration	Include the *force-registration* clause to register the agent with the arguments provided. This clause is useful if you're overriding an existing agent configuration. The default value is *Yes*.
--group	The name of a group in which to place the agent. This parameter is optional. If omitted, the agent isn't placed in a group.
--team	The name of a database role on the PEM backend database server. Access to this agent is restricted to only the named role, the owner, and the pem_super_admin role. This parameter is optional. If omitted, no team is assigned, meaning all users can access this agent.
--owner	The name of a database user on the PEM backend database server. This user will be assigned as the owner of the agent. If omitted, the specified *pem-user* is assigned as the owner.
--cluster-name	Specifies the cluster name in Object Explorer to which the agent object will be added. If the cluster does not exist, it will be created automatically. This parameter is optional.

continues on next page

Table 1 – continued from previous page

Option	Description
--profile-id	Specifies the ID of an existing agent profile to assign to this agent. Optional.
--profile-name	Specifies the name of an existing agent profile to assign to this agent. This parameter is optional and will be overridden by *--profile-id* if set.
--allow_server_restart	Allow PEM to restart the monitored server. The default value is *True*.
--allow-batch-probes	Allow PEM to run batch probes on this agent. The default value is *False*.
--batch-script-user	The operating system user to use for executing the batch/shell scripts. The default value is none. If you leave this parameter blank or the specified user doesn't exist, the scripts don't execute.
--enable-heartbeat-connecti	Create a dedicated heartbeat connection between the PEM agent and server to update the active status. The default value is *False*.
--enable-smtp	Allow the PEM agent to send the email on behalf of the PEM server. The default value is *False*.
--enable-snmp	Allow the PEM agent to send the SNMP traps on behalf of the PEM server. The default value is *False*.
--enable-webhook	Allow the PEM agent to call webhook endpoints on behalf of the PEM server. The default value is *False*.
-o	Override the configuration file options. See the following *Overriding default configurations - examples* for usage.

Note

If you remove *pem-agent-user* parameter from the *agent.cfg* file manually, then the agent will not connect using the default users.

16.2.3 Authenticating the pemworker utility

Before any changes are made on the PEM database, the connection is authenticated with the PEM database server. When invoking the pemworker utility, you must provide the password associated with the PEM server administrative user role (postgres). You can specify the administrative password in three ways:

- Set the `PEM_SERVER_PASSWORD` environment variable.
- Provide the password on the command line with the `PGPASSWORD` keyword.
- Create an entry in the `.pgpass` file.
- Include a password, or encrypted password, in `--pem-registration-conn-string`

If you don't provide the password, a password authentication error occurs. After authentication succeeds, you're prompted for any other missing required information. When the registration is complete, the server confirms that the agent was successfully registered.

16.2.4 Enable the PEM Agent service

Your agent is now registered but is not yet running. To start the agent running and make sure it remains running after a system reboot, you must enable the PEM Agent service.

16.2.5 Linux

```
systemctl enable --now pemagent
```

16.2.6 Windows

Locate the pemAgent service in the Windows Service Manager (services.msc), right-click on it and select **Restart**. Ensure the **Startup Type** is set to **Automatic**.

16.3 Verifying PEM agent registration

The `verify-registration` subcommand is a diagnostic utility introduced in PEM 10.4. It allows administrators to confirm that a PEM Agent is properly registered, active in the PEM database, and capable of connecting to the PEM Server.

This command validates the agent's operational status by performing the following checks:

1. **Configuration integrity:** Validates that the local configuration contains a proper and recognizable Agent ID.
2. **Secure connectivity:** Verifies the ability to establish a connection to the PEM server using the certificate-based authentication configured in the `agent.cfg` file.
3. **Database status:** Confirms that the agent is registered and explicitly marked as **active** in the PEM database.

This subcommand is particularly useful for:

- **Post-registration validation:** Validating agent setup immediately after initial registration.
- **Connectivity troubleshooting:** Diagnosing connectivity issues between the agent and the PEM server.
- **SSL/certificate audits:** Verifying that SSL certificates are properly configured and valid.
- **Health monitoring:** Confirming agent status in automated health checks and monitoring scripts.

16.3.1 Command syntax

The syntax for the `verify-registration` subcommand is specific to each operating system:

Linux

```
pemworker --verify-registration [verify-registration-options]
```

Windows

```
pemworker VERIFY-REGISTRATION [verify-registration-options]
```

Option	Description
--config-dir <path>	The directory path where the agent configuration file (*agent.cfg*) is located. If not specified, the default configuration directory is used (e.g., */usr/edb/pem/agent/etc/* on Linux).

When the command is executed, the `pemworker` utility performs a four-step validation process:

1. **Load configuration:** Reads the `agent.cfg` (from the default or specified path) to retrieve the Agent ID and connection string.

2. **Establish connection:** Attempts to open a connection to the PEM database server using the certificate-based SSL authentication defined during registration.
3. **Query status:** Executes a check against the `pem.agent` system table to verify the ID exists and the status is set to active.
4. **Report result:** Outputs the success or failure message to the console and returns a specific exit code.

16.3.2 Understanding exit codes

The command returns the following exit codes:

Code	Status	Description
0	Success	Agent is registered, active, and connected.
1	Config Error	Failed to retrieve a valid Agent ID from the local configuration.
2	Connection Error	Unable to establish a connection to the PEM Server.
3	Database Error	Agent ID was found, but it is marked as inactive in the PEM database.

16.3.3 Usage examples

Example 1 : Basic Verification

To verify the registration using default settings:

Linux

```
pemworker --verify-registration
```

Windows

```
pemworker VERIFY-REGISTRATION
```

Expected output on success:

```
Verify registration done successfully.
```

Example 2: Using Custom Configuration Directory

To verify an agent that uses a non-default configuration directory:

Linux

```
pemworker --verify-registration --config-dir /opt/custom/pem/agent/etc
```

Windows

```
pemworker VERIFY-REGISTRATION --config-dir "C:\custom\pem\agent\etc"
```

16.4 Unregistering a PEM agent

You can use the pemworker utility to unregister a PEM agent. To unregister an agent, invoke the pemworker utility as shown in the examples that follow.

Linux

```
#  Running as root

pemworker --unregister-agent
```

Windows

```
./pemworker.exe UNREGISTER-AGENT
```

When invoking the pemworker utility, append command line options to the command string. Follow each option with a corresponding value.

Option	Description
--pem-user <username>	The name of the database user (member of pem_admin role) of the PEM backend database server. This parameter is required.
--config-dir	The directory path for the configuration file. The default is *"<pemworker path>/../etc"*.

16.5 Advanced usage

The following are some advanced options for PEM agent registration.

16.5.1 Using connection strings

For finer control over how the PEM agent connects to the PEM backend database, you can specify connection strings. There are two connection strings you can specify, the *registration connection string and the* agent* connection string. These connection strings can be in either key-value or URI format and include any options supported by libpq with the exception of `dbname` which is hardcoded to `pem` and `target_session_attrs` which is hardcoded to `read-write` .

16.5.2 The registration connection string

The registration connection string, specified using `--pem-registration-conn-string` , is used for 'one time' privileged connections to the PEM backend to perform the following tasks.:

- Agent registration (–register-agent)
- Agent unregistration (–unregister-agent)
- Server registration (–register-server)
- Server unregistration (–unregister-server)

The registration string must specify a user with the necessary permissions to perform these actions, specifically

- `pem_admin` role
- `rolcreaterole` unless you also specify an existing user via `--pem-agent-user` or an agent connection string (see below)

The registration connection string itself is never stored, so must be entered whenever you wish to use it to perform one of these tasks. Note that the host and port will be extracted from the connection string used for agent registration and stored in the agent.cfg file so the agent knows where to find the PEM backend.

You can also specify the registration connection string using the environment variable `PEM_REGISTRATION_CONN_STRING` The order of precedence is as follows:

1. Command-line Argument (–pem-registration-conn-string) - Highest priority
2. Environment Variable (PEM_REGISTRATION_CONN_STRING)
3. Individual Parameters (–pem-server, –pem-port, –pem-user) - Lowest priority

16.5.3 The agent connection string

The agent connection string is used for the persistent connection used by the agent to connect to the PEM backend for ongoing data collection, etc. This is an override option, written directly to agent.cfg so must be specified using `-o agent_connection_string` .

When the agent connection string is not set, the agent will use the individual parameters in `agent.cfg` (`pem_host` , `pem_port` , etc.), or the registry on Windows, to determine how and where to connect.

When the agent connection string is set, these parameters are ignored. If the agent connection string is specified during agent registration, PEM will not attempt to create a user or a client certificate for the agent connection. If you supply a connection string, you must ensure that the specified user exists, can authenticate and has the `pem_agent` role.

The agent connection string will be stored in `agent.cfg` , or the registry on Windows. If the string contains a `password` or `sslpassword` value, it will be encrypted and replaced with the custom `pemenc_password` or `pemenc_sslpassword` values respectively. Passwords containing special characters must be enclosed in single quotes (key-value style) or URL encoded (URI style).

You can also specify the agent connection string using the environment variable `PEM_AGENT_CONN_STRING` The order of precedence is as follows:

1. Command-line Argument (`-oagent_conn_string=...`) - Highest priority
2. Environment Variable (`PEM_AGENT_CONN_STRING`)
3. Configuration File/Registry (`agent_conn_string` / `AGENT_CONN_STRING`)
4. Individual Parameters (`pem_host` , `pem_port` , `pem_ssl_mode` , etc.) - Lowest priority

16.5.4 Setting the agent ID

Each registered PEM agent must have a unique agent ID. The value `max(id)+1` is assigned to each agent ID unless a value is provided using the `-o` options as shown *Overriding default configurations - examples* .

16.5.5 Overriding default configurations - examples

The `-o` option allows you to override agent configuration parameters (stored in `agent.cfg` on Linux and `HKEY_LOCAL_MACHINE\Software\EnterpriseDB\PEM\agent` on Windows) that cannot otherwise be set during registration. This option should be used with caution because these inputs will not be validated in the same way as regular CLI options. The full list of parameters is given in *Modifying agent configuration*

Below are some examples of uses of the `-o` option.

16.5.6 Setting the agent ID

Register the PEM agent using the command line. Assign an `agent_id` value of 8 using the `-o` option.

```
#  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!
```

Because the `agent_id` of 8 is available, the PEM agent registers successfully. If the given ID is already in use by an existing agent, an error occurs.

16.5.7 Providing an existing SSL certificate and key

Register the PEM agent using the command line. Assign the existing SSL certificates and key files to avoid generating new ones for a particular agent ID. The SSL certificates and key files must be valid for the database user `agent<ID>` , where `<ID>` must be the same as provided using the command line. Use the `-o` option.

```
#  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 \
- -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!
```

Because the valid SSL certificates and key files are available at the given location with proper permissions, the PEM agent registers successfully. If the certificate or key files aren't valid or don't have proper permissions, an error occurs.

16.5.8 Using a non-root user account to register a PEM agent on Linux

To use a non-root user account to register a PEM agent, you must first install the PEM agent as a root user. After installation, assume the identity of a non-root user, such as edb. Then:

1. Log in as edb. Create `pem` and `logs` directories and assign read, write, and execute permissions:

```
# 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. Register the agent with PEM server:

```
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. Change the parameters of the `agent.cfg` file:

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

Where `<id>` is the assigned PEM agent ID.

4. Create a `tmp` directory, set the environment variable, and start the agent:

```
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
```

Your PEM agent is now registered and started with the edb user. If your machine restarts, then this agent doesn't restart automatically. You need to start it manually using the previous command.

5. Optionally, you can create the service for this PEM agent as the root user to start this agent automatically at machine restart as follows:

a. Update the values for the configuration file path and the user in the `pemagent` service file as superuser:

```
# 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. Stop the running agent process and restart the agent service:

```
# 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
sudo systemctl status pemagent
```

6. Check the agent status on the PEM dashboard.

Note

- Any probes and jobs that require root permission or access to a file owned by another user (for example, enterprisedb) fail. - If you move the *agent.cfg* file from its default location to another, the PEM dashboard might display the agent status as *unknown* . See *Troubleshooting agent issues* for more information.

CHAPTER 17

Registering a Postgres server

Before you can manage or monitor a database server with PEM, you must register the database server with PEM and bind the database server to a previously registered PEM agent. You can bind a database server to a remote agent (an agent that resides on a different host). However, if the agent resides on a different host, it doesn't have access to all of the statistical information about the instance.

17.1 What does registering a Postgres server do?

Registering a server does two things:

- It adds the server to the server tree in the PEM web application, allowing users to open connections from the web application to the server. - It binds the server to a particular agent, meaning that agent will open connections to the server to execute probes and scheduled jobs.

After registration, the agent opens a connection to the specified database. It uses this connection to determine the other databases it can connect to with the same user and then spawns one connection to each.

There are three ways to register a server.

- *Using the pemworker utility to register a server* from the command line on the host where the agent you want to bind is located. - *Registering a server with automatic server discovery* in the PEM web application to automatically register Postgres servers located on the same host as an agent. - *Manually registering a database server* by adding all the details in the PEM web application.

17.2 Using the pemworker utility to register a server

You can use the pemworker utility to register a Postgres server. During registration, the pemworker utility binds the new server to the agent that resides on the system from which you invoked the registration command.

To register a server on a Linux host, use the command:

```
export PEM_SERVER_PASSWORD=example
export PEM_MONITORED_SERVER_PASSWORD=example
pemworker --register-server
```

To register a server on a Windows host, use the command:

```
set PEM_SERVER_PASSWORD=example
set PEM_MONITORED_SERVER_PASSWORD=example
pemworker.exe REGISTER-SERVER
```

The environment variable `PEM_SERVER_PASSWORD` must be set to the password of the specified `--pem-user` . If necessary, you can *Specifying additional parameters for the pemworker connection* for this user. Set the environment variable `PEM_MONITORED_SERVER_PASSWORD` to the password of the Agent-Server binding user (`--asb-host-user` or `--server-user` if the former is not specified).

Append command line options to the command string when invoking the pemworker utility. Follow each option with a corresponding value.

Option	Description
--pem-user	Specifies the name of the PEM administrative user (must have the *pem_admin* role) on the PEM server to use to write the server details to the PEM database. Required.
Server parameters	These parameters are used to populate the connection properties in the PEM web application. They're used when a user connects from the PEM web application to the monitored server. They're also used for connections from the agent to the monitored server unless overridden by Agent Server Binding parameters (see the **Agent-server binding parameters** in this table).
--server-addr	Specifies the IP address or fully qualified domain name of the monitored server. On Linux systems, you can leave the address field blank to use the default PostgreSQL Unix Domain Socket on the local machine. Or you can set it to an alternative path to a PostgreSQL socket. If you enter a path, the path must begin with a forward slash (/). In this case you must still set *PEM_MONITORED_SERVER_PASSWORD* to a non-empty value, although it isn't used. Because sockets are only accessible locally, if you want to use a socket connection, we recommend that you specify this via *asb-host-name* instead. Required.
--server-port	Specifies the port number of the monitored server. Required.
--server-database	Specifies the name of the database on the monitored server to which to connect. Required.
--server-user	Specifies the name of the user used to connect to the monitored server. Required.
--server-service-name	Specifies the name of the operating system service that manages the monitored Postgres server, for example, a systemd service unit on Linux. Optional.
Agent-server binding parameters	Use these parameters to override the server parameters. Use these if you want the agent to connect to the monitored server using different credentials from those used by the PEM web application.
--asb-host-name	Specifies the IP address or fully qualified domain name of the monitored server. Alternatively, on Linux systems, you can provide path to a PostgreSQL socket. If you enter a path, the path must begin with a forward slash (/). In this case you must still set *PEM_MONITORED_SERVER_PASSWORD* to a non-empty value, although it isn't used. Optional, defaults to *--server-addr* if not supplied.
--asb-host-port	Specifies the port number of the monitored server. Optional, defaults to *--server-port* if not supplied.
--asb-host-db	Specifies the name of the database on the monitored server to which the agent connects. Optional, defaults to *--server-database* if not supplied.
--asb-host-user	Specifies the name of the user used by the agent to connect to the monitored server. Optional, defaults to *--server-user* if not supplied.
--asb-ssl-mode	Specifies the type of SSL authentication to use for connections. Supported values include: *prefer*, *require*, *disable*, *verify-CA*, and *verify-full*. Optional, defaults to *prefer*.
Server metadata	These parameters determine how the server is shown in the PEM web application.

continues on next page

Table 1 – continued from previous page

Option	Description
--group	Specifies the name of the group in which the server is displayed. Optional.
--team	Specifies a Postgres role on the PEM server to be assigned as the team to which the monitored server belongs. Only users with this role can access the server. Optional, defaults to none, meaning all users can access.
--owner	Specifies a Postgres user on the PEM server to assign as the owner of the monitored server. Optional, defaults to *--pem-user*.
--display-name <name>	Specifies the display name of the monitored database server. Optional, defaults to the system hostname.
--cluster-name	Specifies the cluster name in Object Explorer to which the server object will be added. If the cluster does not exist, it will be created automatically. Optional.
--profile-id	Specifies the ID of an existing server profile to assign to this server. Optional.
--profile-name	Specifies the name of an existing server profile to assign to this server. This parameter is optional and will be overridden by *--profile-id* if set.
Cluster management parameters	These parameters allow you to pass information about a cluster manager—EDB Failover Manager (EFM) or Patroni—if one is in use.
--replication-solution	Specifies the cluster manager in use with this cluster. Possible values are *efm* and *patroni*. Optional, will be set automatically if the *--efm-* * or `--patroni-* ` options are specified.
--efm-cluster-name	Specifies the name of the EDB Failover Manager cluster that monitors the server (if applicable).
--efm-install-path	Specifies the complete path to the installation directory of EDB Failover Manager (if applicable).
--patroni-cluster-name	Specifies the name of the Patroni cluster of which the server is a member (if applicable).
--patroni-install-path	Specifies the complete path to the installation directory of Patroni (if applicable).
--patroni-config-path	Specifies the complete path to the configuration file of the Patroni cluster (if applicable).
Other parameters	
--remote-monitoring	Set to *yes* if the server isn't located on the same host as the agent. When remote monitoring is enabled (*yes*), agent-level statistics for the monitored server aren't available for custom charts and dashboards, and the remote server isn't accessible by some PEM utilities (such as Audit Manager, Capacity Manager, Log Manager). Optional, defaults to *no*.
--config-dir	Specifies the directory path of the agent configuration file. Optional, defaults to *<pemworker path>/../etc*.
--force-registration	If supplied, registration will go ahead even if an *agent.cfg* file already exists. This option should only be used if directed by EDB support, as it can cause data integrity issues if used improperly.

17.2.1 Specifying additional parameters for the pemworker connection

When registering or unregistering a database server, you must specify an administrative PEM user (`--pem-user`) to perform the registration. You must also specify this user's password as `PEM_SERVER_PASSWORD` .

If you need to specify additional connection parameters, or use a non-password authentication method, follow the instructions below.

The pemworker utility uses libpq to connect to Postgres. As such, you can set many connection properties using libpq environment variables .

For example, if you wish to specify the SSL mode of the connection used to register the agent

```
export PEM_SERVER_PASSWORD=edb
export PGSSLMODE=require
pemworker --register-agent ...
```

Similarly, if you wish to connect using a certificate-authenticated account, you can provide details of the certificate and key. Note you must still provide `PEM_SERVER_PASSWORD` but it can be set to any value as it will not be used.

```
export PGSSLCERT=pem-server.crt
export PGSSLKEY=pem-server.key
export PEM_SERVER_PASSWORD=none
pemworker --register-agent ...
```

17.2.2 Examples

This example registers a server using only the required parameters. The user `admin01` will be used to connect to the PEM server. The credentials supplied for the `--server-*` parameters will be used to create a connection from the agent to the monitored server. The same details will also be used to populate the server connection details in the PEM web application.

```
pemworker --register-server \
          --pem-user admin01 \
          --server-addr pg123.prod.infra.business \
          --server-port 5432 \
          --server-database postgres \
          --server-user prod_admin
```

The following example specifies different parameters to used for the connection from the agent to the monitored server. They override the fully qualified domain name provided in `--server-addr` with `localhost` instead. They also override the user to use a `local_monitor` user. This might be a user who's permitted to connect only from the same host, for example.

This example also specifies a service name, `postgresql` , which means the PEM agent can restart this server to apply configuration changes.

```
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
```

17.2.3 Using the pemworker utility to unregister a server

You can use the pemworker utility to unregister a database server. To unregister a server, invoke the pemworker utility.

On a Linux host, use the command:

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

On a Windows host, use the command:

```
pemworker.exe UNREGISTER-SERVER
```

Append command line options to the command string when invoking the pemworker utility. Follow each option with by a corresponding value:

Option	Description
--pem-user	Specifies the name of the PEM administrative user. Required.
--asb-host-name	Specifies the hostname that the agent will use to connect to the database server.
--server-port	Specifies the port number of the host. Required.
--config-dir	Specifies the directory path of the agent configuration file. Optional, defaults to *<pem-worker path>/../etc*.

Use the `PEM_SERVER_PASSWORD` environment variable to provide the password for the PEM server to allow the pemworker utility to connect to the PEM server as `--pem-user` .

17.3 Registering a server with automatic server discovery

If the server you want to monitor resides on the same host as the monitoring agent, you can use the Auto Discovery dialog box to simplify the registration and binding process.

To enable auto discovery for a specific agent, you must enable the Server Auto Discovery probe. To do so, select the PEM agent in the PEM client tree, and select **Management > Manage Probes**. When the **Manage Probes** tab opens, confirm that the slider in the **Enabled?** column is set to **Yes**.

To open the Auto Discovery dialog box, select a PEM agent in the PEM client tree and select **Management > Auto Discovery**.

When the Auto Discovery dialog box opens, the **Discovered Database Servers** box displays a list of servers that currently aren't being monitored by a PEM agent. Select the box next to a server name to display information about the server in the **Server Connection Details** box and connection properties for the agent in the **Agent Connection Details** box.

Use **Check All** to select the box next to all of the displayed servers or **Uncheck All** to clear all of the boxes to the left of the server names.

The fields in the **Server Connection Details** box provide information about the server that PEM monitors:

- Accept or modify the name of the monitored server in the **Name** field. The specified name is displayed in the tree of the PEM client.
- Use the **Server group** list to select the server group under which the server is displayed in the PEM client tree.
- Use the **Host name/address** field to specify the IP address of the monitored server.
- The **Port** field displays the port that's monitored by the server. You can't modify this field.
- Provide the name of the service in the **Service ID** field. You must provide the service name to enable some PEM functionality.
- The **Maintenance database** field specifies the database to use for the initial connection and any global maintenance operations. Customize the content of the **Maintenance database** field for your installation.

The fields in the **Agent Connection Details** box specify the properties for the PEM agent to use when connecting to the server:

- The **Host** field displays the IP address used for the PEM agent binding.
- The **User name** field displays the name used by the PEM agent when connecting to the selected server.
- The **Password** field displays the password associated with the specified user name.
- Use the **SSL mode** field to specify your SSL connection preferences.

After you finish specifying the connection properties for the servers that you're binding for monitoring, select **OK** to register the servers.

After selecting **OK**, the newly registered server is displayed in the PEM tree and is monitored by the PEM server.

17.4 Manually registering a database server

To manage or monitor a database server with PEM, you must:

- Register your EDB Postgres Advanced Server or PostgreSQL server with the PEM server.
- Bind the database server to a PEM agent.

You can use the Register Server dialog box to provide registration information for a server, bind a PEM agent, and display the server in the PEM client tree. To open the Create Server dialog box, select **Object > Register > Server**.

Note

- You must ensure the *pg_hba.conf* file of the Postgres server that you're registering allows connections from the host of the PEM client before attempting to connect. - Only database superusers or users with the *pem_admin* role can bind a database server to a PEM agent.

Use the **General** tab to describe the general properties of the server:

- Use the **Name** field to specify a name for the server. The name identifies the server in the PEM browser tree.
- You can use **Server group/Cluster** to organize your servers, agents, and clusters in the tree. Using Server groups/Cluster can help you manage large numbers of servers more easily. For example, you can have a production group, a test group, or LAN-specific groups. Use the **Server group/Cluster** list to select the server group/cluster in which to display the new server.
- Use the **Team** field to specify a PostgreSQL role name. Only PEM users who are members of this role, who created the server initially, or have superuser privileges on the PEM server see this server when they log in to PEM. If this field is left blank, by default all PEM users see the server. You can use the `show_objects_with_no_team` parameter in the Server Configuration dialog box to change the behavior. If `show_objects_with_no_team` is set to `false` , the server with no team isn't visible to all other users.
- Use the **Background** color selector to select the color to display in the PEM tree behind database objects that are stored on the server.
- Use the **Foreground** color selector to select the font color of labels in the PEM tree for objects stored on the server.
- Select **Connect now?** to attempt a server connection when you select **Save**. Clear **Connect now?** if you don't want the PEM client to validate the specified connection parameters until a later connection attempt.
- Provide notes about the server in the **Comments** field.

Note

The **Team** field in the Server Properties dialog is editable only if one of the following conditions is met:

- User has the `pem_super_admin` role.
- User is the owner of the server.
- User is creating a new server (this requires the `pem_database_server_registration` role).

In all other scenarios, the field is read-only.

- Specify the IP address of the server host or the fully qualified domain name in the **Host name/address** field. On Unix-based systems, leave the address field blank to use the default PostgreSQL Unix Domain Socket on the local machine. Or you can set an alternative path containing a PostgreSQL socket. If you enter a path, the path must begin with a forward slash (/).
- Specify the port number of the host in the **Port** field.
- Use the **Maintenance database** field to specify the name of the initial database for PEM to connect to that's expected to contain pgAgent schema and adminpack objects installed (both optional). On PostgreSQL version 8.1 and later, the maintenance DB is normally called `postgres` . On earlier versions `template1` is often used, although it's better to create a `postgres` database to avoid cluttering the template database.
- Set **Kerberos Authentication** to **Yes** to use the Kerberos authentication for a monitored server. By default, the monitored server uses the same authentication method as the PEM server. If the monitored server doesn't want to use Kerberos authentication, then set `ALLOW_DATABASE_CONNECTION_WITHOUT_KERBEROS` to `TRUE` in the `config_local.py` file.
- Specify the name to use when authenticating with the server in the **Username** field.
- Provide the password associated with the specified user in the **Password** field.
- Select **Save password?** to store passwords in encrypted format in a PEM backend database for later reuse. Each password is stored on a per-user, per-server basis and isn't shared with other team members. PEM uses the saved password to connect the database server next time. To remove a saved password, disconnect the database server first, and then select **Object > Clear Saved Password**.
- Use the **Role** field to specify the name of the role that's assigned the privileges for the client to use after connecting to the server. This value allows you to connect as one role and then assume the permissions of another role (the one you specified in this field) when the connection is established. The connecting role must be a member of the role specified.

Use the fields in the **Parameters** tab to configure your connection settings:

Click the + button to add a new parameter. Common parameters include:

- **Host address**: Specify the server's IP address to avoid DNS lookups and improve connection speed. When using Kerberos, GSSAPI, SSPI, or verify-full SSL mode, it's recommended to provide both the hostname and IP address.
- **Password file**: Specify the path to a .pgpass file to enable passwordless authentication. For more information, see PostgreSQL documentation, Section 33.15.
- **Connection timeout**: Set the maximum time (in seconds) to wait for a connection. A value of 0 or empty means wait indefinitely. The default is 10 seconds; using a value less than 2 seconds is not recommended.
- **SSL mode**: Select the type of SSL connection to use. For more information, see

the PostgreSQL documentation .

You can use the platform-specific file manager dialog box to upload files that support SSL encryption to the server. To open the file manager, select the icon located to the right of each of the following fields:

- Use the **Client certificate** field to specify the file containing the client SSL certificate. This file replaces the default `<STORAGE_DIR>/<USERNAME>/.postgresql/postgresql.crt` . This parameter is ignored if an SSL connection isn't made.
- Use the **Client certificate key** field to specify the file containing the secret key used for the client certificate. This file replaces the default `<STORAGE_DIR>/<USERNAME>/.postgresql/postgresql.key` . This parameter is ignored if an SSL connection isn't made.
- Use the **Root certificate** field to specify the file containing the SSL certificate authority. This file replaces the default `~/.postgresql/root.crt` file. This parameter is ignored if an SSL connection isn't made.

- Use the **Certificate revocation list** field to specify the file containing the SSL certificate revocation list. This list replaces the default list, found in `~/.postgresql/root.crl` . This parameter is ignored if an SSL connection isn't made.
- When **SSL compression?** is set to **True**, data sent over SSL connections is compressed. The default value is **False** (compression is disabled). This parameter is ignored if an SSL connection isn't made.

> **Warning**
>
> Certificates, private keys, and the revocation list are stored in the per-user file storage area on the server, which is owned by the user account under which the PEM server process is run. This means that administrators of the server might be able to access those files. Use caution before using this feature.

Use the **SSH Tunnel** tab to configure SSH tunneling. You can use a tunnel to connect a database server through an intermediary proxy host to a server that resides on a network to which the client might not be able to connect directly.

- Set **Use SSH tunneling** to **Yes** to use an SSH tunnel when connecting to the specified server.
- Specify the name or IP address of the SSH host (through which client connections are forwarded) in the **Tunnel host** field.
- Specify the port of the SSH host through which client connections are forwarded in the **Tunnel port** field.
- Specify the name of a user with login privileges for the SSH host in the **Username** field.
- Specify the type of authentication to use when connecting to the SSH host in the **Authentication** field.
- Select **Password** to use a password for authentication to the SSH host. This is the default.
- Select **Identity file** to use a private key file when connecting.
- If the SSH host is expecting a private key file for authentication, use the **Identity file** field to specify the location of the key file.
- If the SSH host is expecting a password, use the **Password** field to specify the password. If an identity file is being used, specify the passphrase.
- Enable the **Save password?** to instruct the PEM to save the password for future use. Use Clear SSH Tunnel Password to remove the saved password.
- Specify the number of seconds in the **Keep alive (seconds)** field to define the period in which, if no data was sent over the connection, `keepalive` packet will be sent (ignored by the remote host). This is useful to keep the connections alive over a NAT. Specify 0 to disable `Keep alive` .

Use the **Advanced** tab to specify details that are used to manage the server:

- Use the **DB restriction** field to specify a SQL restriction to use against the pg_database table to limit the databases displayed in the tree. For example, you might enter: `'live_db'` , `'test_db'` to display only the `live_db` and `test_db` databases. You can also limit the schemas shown in the database from the database Properties dialog box by entering a restriction against pg_namespace .
- Use the **Password exec command** field to specify a shell command that retrieves the SQL password at runtime. The command's stdout is used as the password.

This is useful when passwords are short-lived tokens, such as in PAM authentication scenarios.

You can include placeholders in the command to dynamically pass connection details:

- **%HOSTNAME%** – Server hostname
- **%PORT%** – Server port
- **%USERNAME%** – Database username

- Use the **Password exec expiration** field to set the maximum age (in seconds) of the password retrieved by the **Password exec command**.
- If not specified, the password remains valid for the duration of the session.
- If set to 0, the command will be executed for every new connection or reconnection.
- If the password is time-limited (e.g., a temporary token), set this value slightly before its expiration time to avoid authentication failures.
- Use the **Prepare threshold** field to control when queries are prepared:
- **0**: Prepare the query on its first execution.
- **Positive integer**: Prepare the query after it has been executed that many times.
- **Blank**: Disable prepared statements entirely for this connection.

This setting is especially useful when using external connection pooling tools like PgBouncer, which are incompatible with prepared statements — set this field to blank in such cases.

- Use the **Service ID** field to specify parameters to the database service process. For servers that are stored in the Enterprise Manager directory, enter the service ID. On Windows machines, this is the identifier for the Windows service. On Linux machines, the name of the init script used to start the server is `/etc/init.d` and the name of the systemd script to start the server is `systemctl` . For example, the name of the EDB Postgres Advanced Server 11 service is `edb-as-11` . For local servers, the setting is operating system dependent:
- If the PEM client is running on a Windows machine, it can control the postmaster service if you have enough access rights. Enter the name of the service. In case of a remote server, prepend it with the machine name (such as `PSE1\pgsql-8.0`). PEM automatically discovers services running on your local machine.
- If the PEM client is running on a Linux machine, it can control processes running on the local machine if you have enough access rights. Provide a full path and needed options to access the `pg_ctl` program. When executing service control functions, PEM appends status/start/stop keywords to this. For example:

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

where `x` is the version of the PostgreSQL database server.

Use the **Post Connection SQL** tab to specify SQL queries.

- Use the **Post Connection SQL** field to write the SQL queries that executes in autocommit mode for each connection made to any of the database in the server.

Use the **Tags** tab to add tags.

- Use the **Tags** tab to add custom tags, that appears next to the server node label in the Object Explorer tree.

To add a tag: - Click the + button. - Use the **Text** field to specify the tag name. - Use the **Color** field to select an accent color for the tag.

Use the **Replication** tab to select the replication solution used by the server.

- Use the **Replication Solution** for your server:
- If the server is a member of any replication solution, you can use PEM to monitor the health of the cluster and to replace the primary node if necessary. **EFM**
- To enable PEM to monitor Failover Manager, use the **EFM cluster name** field to specify the cluster name. The cluster name is the prefix of the name of the Failover Manager cluster properties file name. For example, if the file name is `efm.properties` , the cluster name is efm.
- If you're using PEM to monitor the status of a Failover Manager cluster, use the **EFM installation path** field to specify the location of the Failover Manager binary file. By default, the Failover Manager binary file is installed in `/usr/edb/efm-x.x/bin` , where `x.x` specifies the Failover Manager version. **Patroni**

- To enable PEM to monitor Patroni cluster, use the **Patroni cluster name** field to specify the cluster name. The default name of the cluster is Patroni. You can find the cluster name in the `Scope` value of the Patroni configuration file. The configuration file is in YAML format and often named as patroni.yaml.
- Use the **Patroni installation path** to specify the Patroni binary file location. The common locations are: - `/usr/local/bin` - `/usr/bin`
- Use the **Patroni config path** field to specify the configuration file. The common locations are: - `/etc/patroni/patroni.yaml` - `/etc/patroni.yaml` - The directory where the Patroni process is started **None**
- Select **None** if you are not using any of the replication solution.

Use the **PEM Agent** tab to specify connection details for the PEM agent.

On the **Connection Parameters** tab:

- Use the **Bound agent** list to select a PEM agent. One agent can monitor multiple Postgres servers.
- Set **Remote monitoring?** to **Yes** to indicate that the PEM agent doesn't reside on the same host as the monitored server. When remote monitoring is enabled, agent level statistics for the monitored server aren't available for custom charts and dashboards, and the remote server can't be accessed by some PEM utilities (such as Audit Manager, Capacity Manager, Log Manager).
- In the **Host** field, enter the IP address or socket path for the agent to use when connecting to the database server. By default, the agent uses the host address shown on the **General** tab. On a Unix server, you might want to specify a socket path, such as `/tmp` .
- Enter the **Port** number for the agent to use when connecting to the server. By default, the agent uses the port defined on the **Properties** tab.
- Use the **SSL** field to specify an SSL operational mode. Specify **require**, **prefer**, **allow**, **disable**, **verify-ca**, or **verify-full**.

Mode	Description
require	Require SSL encryption for transactions between the server and the agent.
prefer	Use SSL encryption between the server and the agent if SSL encryption is available.
allow	Allow the connection to use SSL if required by the server.
disable	Disable SSL encryption between the agent and the server.
verify-ca	Require SSL encryption and require the server to authenticate using a certificate registered by a certificate authority.
verify-full	Require SSL encryption and require the server to authenticate using a certificate registered by a trusted certificate authority.

For more information about using SSL encryption, see the PostgreSQL documentation .

- Use the **Database** field to specify the name of the Postgres database to which the agent initially connects.
  - If you're registering a EDB Postgres Distributed database node then specify the EDB Postgres Distributed-enabled database name in this field. In the **Username** field, specify the name of the role for the agent to use when connecting to the server. The specified role must be a database superuser for all of the features to work as expected. For the list of features that don't work if the specified role isn't a database superuser, see *Setting agent privileges* . If you're using the Kerberos authentication method, then specify the user having the pgd_monitor or pgd_superuser role in this field.
- If you're using Postgres version 10 or later, you can use the pg_monitor role to grant the required privileges to a non-superuser. For information about the pg_monitor role, see Default Roles . In the **Username** field, specify the name of the user for the agent to use when connecting to the server. The specified role must be a database superuser for all of the features to work as expected. If you're using Postgres version 10 or later, you can use the

pg_monitor role to grant the required privileges to a nonsuperuser. For information about the pg_monitor role, see Default Roles .

- Specify the password for the agent to use when connecting to the server in the **Password** field and **Confirm password** fields. If you don't specify a password, you must configure the authentication for the agent manually. For example, you can use a `.pgpass` file, which must be present and accessible on the system where agent is installed.

On the **Advanced** tab:

- Set **Allow takeover?** to **Yes** to specify that another agent can take over the server. This feature allows an agent to take responsibility for the monitoring of the database server if, for example, the server moved to another host as part of a high-availability failover process.
- Use the plus sign (+) to add the database you want to exclude from the PEM monitoring. You can't exclude the database mentioned on the **Connection Parameters** tab of the **PEM Agent** tab.

Note

The database-level probes don't execute for excluded databases, but the server-level probes can collect the database statistics.

To view the properties of a server, right-click the server name in the PEM client tree, and select **Properties** from the context menu. To modify a server's properties, disconnect from the server before opening the Properties dialog box.

17.5 Verifying the connection and binding

Once registered, the new server is added to the PEM browser tree and is displayed on the Global Overview dashboard.

When first connecting to a newly bound server, the Global Overview dashboard might display the new server with a status of "unknown" in the server list. Before recognizing the server, the bound agent must execute a number of probes to examine the server. These probes might take a few minutes to complete, depending on network availability.

In a few minutes, bar graphs on the Global Overview dashboard show that the agent connected successfully. The new server is included in the **Postgres Server Status** list.

If after five minutes the Global Overview dashboard still doesn't list the new server, check the logfiles for the monitoring agent for errors. Right-click the agent's name in the tree, and select the **Dashboards > Probe Log Analysis** from the context menu.

CHAPTER 18

Managing a PEM server

Some of the tasks related to managing the PEM server include:

- Restarting the PEM server and agent
- Controlling the PEM server or PEM agent
- Controlling the web server
- Managing PEM authentication and security
- Modifying the `pg_hba.conf` file
- Modifying PEM to use a proxy server

18.1 Starting and stopping the PEM server and agents

The PEM server starts, stops, and restarts when the Postgres server instance where it resides starts, stops, or restarts. Use the same commands to control the PEM server that you use to control the Postgres server. On Linux platforms, the command that stops and starts the service script varies by platform and OS version.

The PEM agent is controlled by a service named pemagent.

The Windows operating system includes a graphical service controller that displays the server status and offers interactive server control. You can access the Services utility through the Windows Control Panel. When the utility opens, navigate through the listed services and select the service name.

Select **Stop**, **Pause**, **Start**, or **Restart** to control the state of the service.

Any user or client application connected to the Postgres server is abruptly disconnected if you stop the service. For more information about controlling a service, see EDB Postgres Advanced Server .

18.2 Controlling the PEM server or PEM agent on Linux

On Linux platforms, the name of the service script that controls:

- A PEM server on EDB Postgres Advanced Server is `edb-as-<x>` or just `edb-as` .
- A PEM server on PostgreSQL is `postgresql-<x>` .
- A PEM agent is `pemagent` .

Where `x` indicates the server version number.

You can use the service script to control the service.

To control a service, at the command prompt, assume superuser privileges and issue the command:

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

Where:

`service_name` is the name of the service.

`action` specifies the action taken by the service. Specify:

- `start` to start the service.
- `stop` to stop the service.
- `restart` to stop and then start the service.
- `status` to check the status of the service.

18.3 Controlling the PEM server or PEM agent on Windows

The Windows operating system includes a graphical service controller that displays the server status and offers interactive server control. The registered name of the service that controls:

- A PEM server host on PostgreSQL is `postgresql-<x>` .
- A PEM server host on Advanced Server is `edb-as-<x>` or `ppas-<x>` .
- A PEM agent is `Postgres Enterprise Manager - pemAgent` .

Where `x` indicates the server version number.

In the Windows Control Panel, open the Services utility to see the list of services.

Select **Stop the service** to stop a service. Any user or client application connected to the server is abruptly disconnected if you stop the service.

Select **Pause the service** to reload a service's configuration parameters. **Pause the service** is an effective way to reset parameters without disrupting user sessions for many of the configuration parameters.

Select **Start the service** to start a service.

18.4 Controlling the web server on Linux

On Linux, both NGINX and Apache HTTPD are supported. The default for new installs is NGINX.

You can check which web server your installation is using by running the command below. If it returns `USE_NGINX=1` then your installation is using NGINX. If it returns `USE_NGINX=0` , or returns nothing, then your installation is using Apache HTTPD.

```
grep USE_NGINX /usr/edb/pem/share/.install-config
```

18.4.1 NGINX

You can check the status of the NGINX service with the following command.

```
systemctl status nginx
```

If the service is not running you can start it.

```
systemctl start nginx
```

When NGINX is used as the web server, PEM uses a separate service `edb-uwsgi` to run the application itself.

You can check the status of the `edb-uwsgi` service with the following command.

```
systemctl status edb-uwsgi
```

If the service is not running you can start it.

```
systemctl start edb-uwsgi
```

18.4.2 Apache HTTPD

You can check the status of the Apache service with the following command, substituting `<service-name>` for the name of the service on you operating system. This is generally `httpd` for RHEL-like systems and `apache2` for Debian-like systems.

```
systemctl status <service-name>
```

If the service is not running you can start it.

```
systemctl start <service-name>
```

18.5 Controlling the web server on Windows

On Windows, the web server is a bundled version of Apache HTTPD called PEM-HTTPD.

You can use the Services applet to check the status of the PEM HTTPD service. After opening the Services applet, select the PEM HTTPD service from the list.

The Status column displays the current state of the server. Select **Start** to start PEM HTTPD if the service isn't running.

18.6 Managing PEM authentication

Postgres supports a number of authentication methods:

- Secure password (md5)
- GSSAPI
- SSPI
- Kerberos
- Ident

- LDAP
- RADIUS
- Certificate (SSL)
- PAM

Postgres and PEM authentication is controlled by the `pg_hba.conf` configuration file. Entries in the configuration file specify:

- Who can connect to a specific database
- The type of authentication required before that user is allowed to connect

A typical entry in the `pg_hba.conf` file that allows a user named postgres to connect to all databases from the local host (127.0.0.1/32) using secure password (md5) authentication connections takes the form:

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

Depending on your system configuration, you might also need to create a password file for the user account that the PEM agent uses to connect to the server. This file allows the agent to properly respond to the server's authentication request. An entry in the password file for a user named postgres, with a password of `1safepwd` takes the form:

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

The password file is usually named `~root/.pgpass` on Linux systems or `%APPDATA%\postgresql\pgpass.conf` on Windows. For more information about configuring a password file, see the PostgreSQL website .

For more information about the authentication methods supported by Postgres, see the PostgreSQL core documentation .

18.7 Editing the PEM server configuration

You can use the PEM client to manage the configuration parameters of the PEM server to enable features or modify default settings. To open the Server Configuration dialog box, select **Management > Server Configuration**. Then edit the parameter values as needed.

18.8 Managing security

PEM provides an interface for managing your Postgres roles and servers.

18.8.1 Login roles

When you connect to the PEM server, you must provide role credentials that allow access to the database on which the PEM server stores data. By default, the postgres superuser account is used to initially connect to the server, but we strongly recommend (for both security and auditing purposes) creating individual roles for each connecting user. You can use the PEM Query tool, the PEM web interface Create – Login/Group Role dialog box, or a command line client (such as psql) to create a role.

To use the Create – Login/Group Role dialog box to create a role:

1. In the PEM tree, expand the node for the server where you want to create the role.
2. Right-click the **Login/Group Roles** node.
3. From the context menu, select **Create > Login/Group Role**.
4. Use the tabs of the Create – Login/Group Role dialog box to define the role.
5. After you finish defining the new role, to create the role, select **Save**.

To modify the properties of an existing login role, right-click the name of a login role in the tree and select **Properties** from the context menu. To delete a login role, right-click the name of the role and select **Delete/Drop** from the context menu.

For more complete information about creating and managing a role, see the PostgreSQL online documentation .

18.8.2 Group roles

Group roles can serve as containers and are used to dispense system privileges, such as creating databases, and object privileges, such as inserting data into a table. The primary purpose of a group role is to make the mass management of system and object permissions easier for a DBA. Rather than assigning or modifying privileges individually across many different login accounts, you can assign or change privileges for a single role and then grant that role to many login roles at once.

Use the **Group Roles** node (located beneath the name of each registered server in the PEM tree) to create and manage group roles. Options on the context menu provide access to a dialog box that allows you to create a new role or modify the properties of an existing role. You can find more information about creating roles in the PostgreSQL documentation .

18.8.3 Using PEM predefined roles to manage access to PEM functionality

You can use the Login/Group Role dialog box to allow a role with limited privileges to access PEM features such as the Audit Manager, Capacity Manager, or SQL Profiler. PEM predefined roles allow access to PEM functionality. Roles that are assigned membership in these roles can access the associated feature.

When defining a user, use the **Membership** tab to specify the roles in which the new user is a member. The new user shares the privileges associated with each role in which it's a member. For a user to have access to PEM extended functionality, the role must be a member of the pem_user role and the predefined role that grants access to the feature. Use the **Roles** field to select predefined role names from a list.

The **SQL** tab displays the SQL command that the server executes when you select **Save**.

This example creates a login role named acctg_clerk that has access to Audit Manager. The role can make unlimited connections to the server at any given time.

You can use PEM predefined roles to allow access to the capabilities listed in the table.

Value	Parent role	Description
pem_super_admin		Role to manage/configure everything on PEM
pem_admin	pem_super_admin	Role for administration/management/configuration of all visible agents/servers and monitored objects
pem_user		Role for having read-only access to all the agents, servers, or monitored objects that are visible to a user having pem_user role. A user with pem_user role can view only those objects where this role has been mentioned in the Team field under the server's properties.
pem_config	pem_admin	Role for configuration management of PEM
pem_component	pem_admin	Role to run/execute all wizard/dialog box-based components
pem_rest_api	pem_admin	Role to access the REST API
pem_server_service_manager	pem_admin	Role for allowing to restart/reload the monitored database server (if server-id provided)
pem_manage_schedule_task	pem_admin	Role to configure the schedule tasks
pem_manage_chart	pem_admin	Role for managing/configuring custom charts.

continues on next page

Table 1 – continued from previous page

Value	Parent role	Description
pem_manage_alert	pem_admin	Role for managing/configuring alerts and its templates
pem_config_alert	pem_config, pem_manage_alert	Role for configuring the alerts on any monitored objects
pem_manage_probe	pem_admin	Role to create, update, and delete the custom probes and change custom probe configuration
pem_config_probe	pem_config, pem_manage_probe	Role for probe configuration (history retention, execution frequency, enable/disable the probe) on all visible monitored objects
pem_database_server_registration	pem_admin	Role to register a database server
pem_comp_auto_discovery	pem_component	Role to run the auto discovery of a database server dialog box
pem_comp_sqlprofiler	pem_component	Role to run the SQL Profiler
pem_manage_efm	pem_admin	Role to manage Failover Manager
pem_comp_capacity_manager	pem_component	Role to run the Capacity Manager
pem_comp_log_manager	pem_component	Role to run the Log Manager
pem_comp_audit_manager	pem_component	Role to run the Audit Manager
pem_comp_bart	pem_component	Role to configure and manage BART server.
pem_comp_performance_diagnostic	pem_component	Role to run the Performance Diagnostics.
pem_query_tool	pem_admin	Role to access the Query tool.
pem_schema_diff_tool	pem_admin	Role to access the Schema Diff tool.
pem_debugger_tool	pem_admin	Role to access the Debugger tool.
pem_erd_tool	pem_admin	Role to access the ERD tool.
pem_grant_wizard	pem_admin	Role to access the Grant wizard.
pem_backup_tool	pem_admin	Role to access the Backup tool.
pem_restore_tool	pem_admin	Role to access the Restore tool.
pem_import_export_data_tool	pem_admin	Role to access the Import/Export Data tool.
pem_search_objects_tool	pem_admin	Role to access the Search Objects tool.
pem_maintenance_tool	pem_admin	Role to access the Maintenance tool.
pem_list_barman	pem_comp_barman, pem_list_all_assets	Role to list the BARMAN server.
pem_list_all_assets	pem_super_admin	Role to view all monitored resources and assets, bypassing any restrictions set by the 'team' field.
pem_manage_assets	pem_admin	Role to add or remove database server, agent, or tool (such as BARMAN).
pem_assign_profile	pem_database_server_registration, pem_manage_profile	Role to assign a profile to agent or server.

18.8.4 Using a team role

When you register a server for monitoring by PEM, you can specify a team to associate with the server. A team is a group role that you can use to allow or restrict access to one or more monitored servers to a limited group of role members. The PEM client displays a server with a specified team only to those users who are:

- A member of the team role

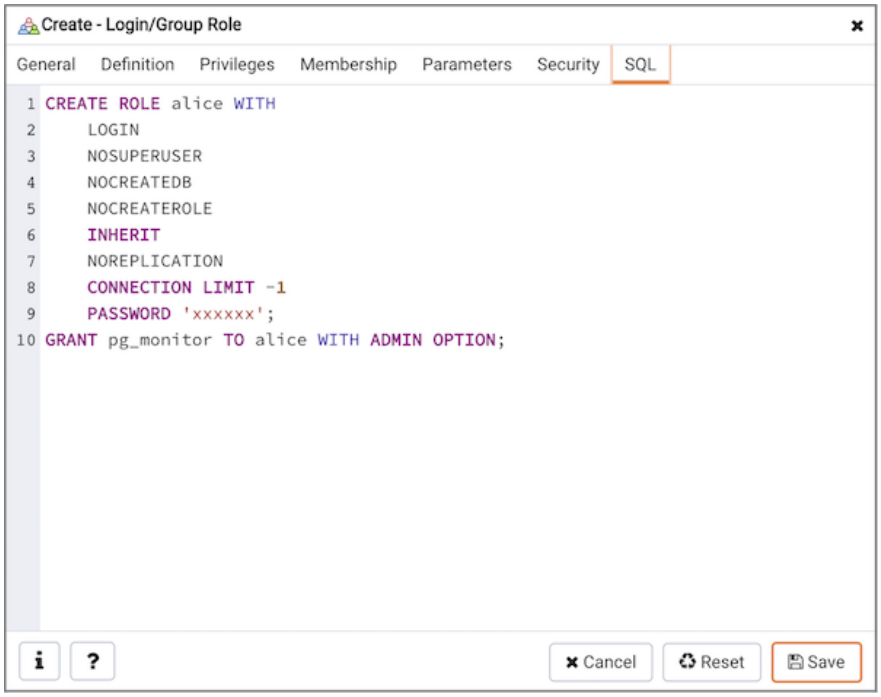


Fig. 1: Create Login Role - SQL tab

- The role that created the server
- A role with superuser privileges on the PEM server

To create a team role:

1. In the PEM tree, expand the node for the server where you want to create the role.
2. Right-click the **Login/Group Roles** node.
3. From the context menu, select **Create > Login/Group Role**.
4. In the Create - Login/Group Role dialog box, use the fields to specify the properties of the team role.

18.8.5 Object permissions

A role must be granted sufficient privileges before accessing, executing, or creating any database object. PEM allows you to assign (`GRANT`) and remove (`REVOKE`) object permissions to group roles or login accounts using the PEM client interface.

Object permissions are managed with the graphical object editor for each particular object. For example, to assign privileges to access a database table, right-click the table name in the tree and select **Properties** from the context menu. Use the **Privileges** tab to assign privileges for the table.

The PEM client also contains a Grant wizard (accessed through the **Tools** menu) that allows you to manage many object permissions at once.

18.9 Server configuration

You can use the Server Configuration dialog box to modify values of user-configurable parameters that control PEM behavior. To access the Server Configuration dialog box, connect to the PEM server, and select **File > Server Configuration**.

Enter a parameter name in the search box in the upper-right corner of the dialog to locate a specific parameter in the list.

To modify a parameter value, edit the content displayed in the **Value** field to the right of a parameter name. To save your changes, select the **Save** icon in the upper-right corner of the dialog box.

18.10 Server configuration parameters - reference

You can use global configuration options to modify aspects of the PEM Server's behavior. The list of configuration parameters is subject to change.

Parameter name	Value and unit	Description
audit_log_retention_time	30 days	Specifies the number of days for an audit log to be retained on the PEM server.
auto_create_agent_alerts	true	Specifies whether to create default agent level alerts automatically when an agent is registered.
auto_create_server_alerts	true	Specifies whether to create default server level alerts automatically when a server is bound to an agent.
chart_disable_bullets	false	Enables/disables bullets on line charts on dashboards and Capacity Manager reports.
cm_data_points_per_report	50	Specifies the number of data points to plot on charts on Capacity Manager reports.
cm_max_end_date_in_ye	5 years	Specifies the maximum amount of time for the Capacity Manager to extrapolate data for. Ensures that threshold-based end dates of reports aren't extrapolated indefinitely.
dash_alerts_timeout	60 seconds	Specifies the number of seconds after which the components of the Alerts dashboard are refreshed.
dash_db_comrol_span	7 days	Specifies the number of days worth of data to plot on the Commit/Rollback Analysis chart on the Database Analysis and Server Analysis dashboards.
dash_db_comrol_timeout	1800 seconds	Specifies the number of seconds after which the Commits/Rollbacks line chart is refreshed on the Database Analysis and Server Analysis dashboards.
dash_db_connovervw_timeout	300 seconds	Specifies the number of seconds after which the Connection Overview pie chart is refreshed on the Database Analysis dashboard.
dash_db_eventlag_span	7 days	Specifies the number of days worth of data to plot on the Number of Events Lag chart for slony replication on the Database Analysis dashboard.
dash_db_eventlag_timeout	1800 seconds	Specifies the number of seconds after which the Number of Events Lag line chart for slony replication is refreshed on the Database Analysis dashboard.
dash_db_hottable_rows	25 rows	Specifies the number of rows to show on the HOT Table Analysis table on the Database Analysis dashboard.
dash_db_hottable_timeout	300 seconds	Specifies the number of seconds after which the Hot Tables table is refreshed on the Database Analysis dashboard.
dash_db_io_span	7 days	Specifies the number of days worth of data to plot on the Database I/O Analysis chart on the Database Analysis and I/O Analysis dashboards.

continues on next page

Table 2 – continued from previous page

Parameter name	Value and unit	Description
dash_db_io_timeout	1800 seconds	Specifies the number of seconds after which the Database I/O line chart is refreshed on the Database Analysis and I/O Analysis dashboards.
dash_db_rowact_span	7 days	Specifies the number of days worth of data to plot on the Row Activity Analysis chart on the Database Analysis, I/O Analysis, and Server Analysis dashboards.
dash_db_rowact_timeout	1800 seconds	Specifies the number of seconds after which the Row Activity line chart is refreshed on the Database Analysis, I/O Analysis, and Server Analysis dashboards.
dash_db_storage_timeout	300 seconds	Specifies the number of seconds after which the Storage bar chart is refreshed on the Database Analysis dashboard.
dash_db_timelag_span	7 days	Specifies the number of days worth of data to plot on the Time Lag chart for Slony replication on the Database Analysis dashboard.
dash_db_timelag_timeout	1800 seconds	Specifies the number of seconds after which the Time Lag line chart for Slony replication is refreshed on the Database Analysis dashboard.
dash_db_useract_span	7 days	Specifies the number of days worth of data to plot on the User Activity Analysis chart on the Database Analysis dashboard.
dash_db_useract_timeout	1800 seconds	Specifies the number of seconds after which the User Activity line chart is refreshed on the Database Analysis dashboard.
dash_efm_timeout	300 seconds	Specifies the number of seconds after which the Failover Manager Node Status and Failover Manager Cluster Info line chart is refreshed on the Streaming Replication dashboard.
dash_global_overview_ti	30 seconds	Specifies the number of seconds after which the components of the Global Overview dashboard are refreshed.
dash_header_timeout	60 seconds	Specifies the number of seconds after which the information on the header of all the dashboards are refreshed.
dash_io_chkpt_span	7 days	Specifies the number of days worth of data to plot on the Checkpoints chart on the I/O Analysis dashboard.
dash_io_chkpt_timeout	1800 seconds	Specifies the number of seconds after which the Checkpoints line chart is refreshed on the I/O Analysis dashboard.
dash_io_hotindx_timeout	300 seconds	Specifies the number of seconds after which the Hot Indexes bar chart is refreshed on the I/O Analysis dashboard.
dash_io_hottbl_timeout	300 seconds	Specifies the number of seconds after which the Hot Tables bar chart is refreshed on the I/O Analysis dashboard.
dash_io_index_objectio_rows	25 rows	Specifies the number of rows displayed on the Index Activity table on the I/O Analysis and Object Activity Analysis dashboards.

continues on next page

Table 2 – continued from previous page

Parameter name	Value and unit	Description
dash_io_index_objectio_timeout	60 seconds	Specifies the number of seconds after which the Index Activity table is refreshed on the I/O Analysis and Object Activity Analysis dashboards.
dash_io_objectio_rows	25 rows	Specifies the number of rows displayed on the Object I/O Details table on the I/O Analysis and Object Activity Analysis dashboards.
dash_io_objectio_timeout	300 seconds	Specifies the number of seconds after which the Object I/O Details table is refreshed on the I/O Analysis and Object Activity Analysis Dashboards.
dash_memory_hostmemact_span	7 days	Specifies the number of days worth of data to plot on the Host Memory Activity Analysis chart on the Memory Analysis dashboard.
dash_memory_hostmemact_timeout	1800 seconds	Specifies the number of seconds after which the Host Memory Activity line chart is refreshed on the Memory Analysis dashboard.
dash_memory_hostmemconf_timeout	300 seconds	Specifies the number of seconds after which the Host Memory Configuration pie chart is refreshed on the Memory Analysis and Server Analysis dashboards.
dash_memory_servmemact_span	7 days	Specifies the number of days worth of data to plot on the server Memory Activity Analysis chart on the Memory Analysis dashboard.
dash_memory_servmemact_timeout	1800 seconds	Specifies the number of seconds after which the Server Memory Activity line chart is refreshed on the Memory Analysis dashboard.
dash_memory_servmemconf_timeout	300 seconds	Specifies the number of seconds after which the Server Memory Configuration pie chart is refreshed on the Memory Analysis dashboard.
dash_objectact_objstorage_rows	15 rows	Specifies the number of rows to show on the Object Storage table on the Object Activity Analysis dashboard.
dash_objectact_objstorage_timeout	300 seconds	Specifies the number of seconds after which the Object Storage table is refreshed on the Object Activity Analysis dashboard.
dash_objectact_objtopindexes_timeout	300 seconds	Specifies the number of seconds after which the Top 5 Largest Indexes bar chart is refreshed on the Object Activity Analysis dashboard.
dash_objectact_objtoptables_timeout	300 seconds	Specifies the number of seconds after which the Top 5 Largest Tables bar chart is refreshed on the Object Activity Analysis dashboard.
dash_os_cpu_span	7 days	Specifies the number of days worth of data to plot on the CPU chart on the Operating System Analysis dashboard.
dash_os_cpu_timeout	1800 seconds	Specifies the number of seconds after which the CPU line chart is refreshed on the Operating System Analysis dashboard.
dash_os_data_span	7 days	Specifies the number of days worth of data to plot on the I/O line chart on the Operating System Analysis dashboard.

continues on next page

Table 2 – continued from previous page

Parameter name	Value and unit	Description
dash_os_disk_span	7 days	Specifies the number of days worth of data to plot on the Utilisation chart on the Operating System Analysis dashboard.
dash_os_hostfs_timeout	1800 seconds	Specifies the number of seconds after which the Host File System Details table is refreshed on the Operating System Analysis dashboard.
dash_os_io_timeout	1800 seconds	Specifies the number of seconds after which the I/O line chart is refreshed on the Operating System Analysis dashboard.
dash_os_memory_span	7 days	Specifies the number of days worth of data to plot on the Memory chart on the Operating System Analysis dashboard.
dash_os_memory_timeout	1800 seconds	Specifies the number of seconds after which the Memory line chart is refreshed on the Operating System Analysis dashboard.
dash_os_packet_span	7 days	Specifies the number of days worth of data to plot on the Packet chart on the Operating System Analysis dashboard.
dash_os_packet_timeout	1800 seconds	Specifies the number of seconds after which the Network Packets line chart is refreshed on the Operating System Analysis dashboard.
dash_os_process_span	7 days	Specifies the number of days worth of data to plot on the Process chart on the Operating System Analysis dashboard.
dash_os_process_timeout	1800 seconds	Specifies the number of seconds after which the Process line chart is refreshed on the Operating System Analysis dashboard.
dash_os_storage_timeout	1800 seconds	Specifies the number of seconds after which the Storage pie chart is refreshed on the Operating System Analysis dashboard.
dash_os_traffic_span	7 days	Specifies the number of days worth of data to plot on the Traffic chart on the Operating System Analysis dashboard.
dash_os_traffic_timeout	1800 seconds	Specifies the number of seconds after which the Traffic line chart is refreshed on the Operating System Analysis dashboard.
dash_os_util_timeout	1800 seconds	Specifies the number of seconds after which the Utilization line chart is refreshed on the Operating System Analysis dashboard.
dash_patroni_timeout	300 seconds	Specifies the refresh interval, in seconds, for the Patroni Node Status and Patroni Cluster Status line charts on the Streaming Replication dashboard.
dash_probe_log_timeout	300 seconds	Specifies the number of seconds after which the Probe Log table refreshed.
dash_replication_archivestat_span	7 days	Specifies the number of days worth of data to plot on the WAL Archive Status chart on the Streaming Replication Analysis dashboard.
dash_replication_archivestat_timeout	1800 seconds	Specifies the number of seconds after which the WAL Archive Status line chart is refreshed on the Streaming Replication dashboard.

continues on next page

Table 2 – continued from previous page

Parameter name	Value and unit	Description
dash_replication_page-lag_span	7 days	Specifies the number of days worth of data to plot on the WAL Lag Pages chart on the Streaming Replication dashboard.
dash_replication_page-lag_timeout	1800 seconds	Specifies the number of seconds after which the WAL Lag Pages line chart is refreshed on the Streaming Replication dashboard.
dash_replication_seg-mentlag_span	7 days	Specifies the number of days worth of data to plot on the WAL Lag Segments chart on the Streaming Replication dashboard.
dash_replication_seg-mentlag_timeout	1800 seconds	Specifies the number of seconds after which the WAL Lag Segments line chart is refreshed on the Streaming Replication dashboard.
dash_replication_time-lag_span	7 days	Specifies the number of days worth of data to plot on the Replication Lag Time chart on the Streaming Replication dashboard.
dash_replication_time-lag_timeout	1800 seconds	Specifies the number of seconds after which the Replication Lag Time line chart is refreshed on the Streaming Replication dashboard.
dash_server_buffers_written	168 hours	Specifies the number of days worth of data to plot on the Background Writer Statistics chart on the Server Analysis dashboard.
dash_server_buffers_written_timeout	300 seconds	Specifies the number of seconds after which the Background Writer Statistics line chart is refreshed on the Server Analysis dashboard.
dash_server_con-novervw_timeout	300 seconds	Specifies the number of seconds after which the Connection Overview pie chart is refreshed on the Server Analysis dashboard.
dash_server_database_tin out	300 seconds	Specifies the number of seconds after which the Databases table is refreshed on the Server Analysis dashboard.
dash_server_db-size_span	7 days	Specifies the number of days worth of data to plot on the Database Size Analysis on the Server Analysis dashboard.
dash_server_db-size_timeout	1800 seconds	Specifies the number of seconds after which the Database Size line chart is refreshed on the Server Analysis dashboard.
dash_server_disk_time-out	1800 seconds	Specifies the number of seconds after which the Disk line chart is refreshed on the Server Analysis dashboard.
dash_server_global_span	7 days	Specifies the number of days worth of data to plot on the Disk line chart on the Server Analysis dashboard.
dash_server_shared-buff_span	7 days	Specifies the number of days worth of data to plot on the Shared Buffer chart on the Server Analysis dashboard.
dash_server_shared-buff_timeout	1800 seconds	Specifies the number of seconds after which the Shared Buffers line chart is refreshed on the Server Analysis dashboard.
dash_server_tabspace-size_span	7 days	Specifies the number of days worth of data to plot on the Tablespace Size chart on the Server Analysis dashboard.

continues on next page

Table 2 – continued from previous page

Parameter name	Value and unit	Description
dash_server_tabspace-size_timeout	1800 seconds	Specifies the number of seconds after which the Tablespace Size line chart is refreshed on the Server Analysis dashboard.
dash_server_user-act_span	7 days	Specifies the number of days worth of data to plot on the User Activity chart on the Server Analysis dashboard.
dash_server_user-act_timeout	1800 seconds	Specifies the number of seconds after which the User Activity line chart is refreshed on the Server Analysis dashboard.
dash_sess_waits_time-wait_timeout	300 seconds	Specifies the number of seconds after which the Session Waits By Time Waited pie chart is refreshed on the Session Waits Analysis dashboard.
dash_sess_waits_wait-dtl_timeout	300 seconds	Specifies the number of seconds after which the Session Waits Details table is refreshed on the Session Waits Analysis dashboard.
dash_sessact_lock-act_timeout	300 seconds	Specifies the number of seconds after which the Session Lock Activity table is refreshed on the Session Activity Analysis dashboard.
dash_sessact_work-load_timeout	300 seconds	Specifies the number of seconds after which the Session Workload table is refreshed on the Session Activity Analysis dashboard.
dash_storage_db-dtls_timeout	300 seconds	Specifies the number of seconds after which the Database Details table is refreshed on the Storage Analysis dashboard.
dash_stor-age_dbovervw_timeout	300 seconds	Specifies the number of seconds after which the Database Overview pie chart is refreshed on the Storage Analysis dashboard.
dash_storage_host-dtls_timeout	300 seconds	Specifies the number of seconds after which the Host Details table is refreshed.
dash_storage_hos-tovervw_timeout	300 seconds	Specifies the number of seconds after which the Host Overview pie chart is refreshed on the Storage Analysis dashboard.
dash_stor-age_tblspcdtls_timeout	300 seconds	Specifies the number of seconds after which the Tablespace Details table is refreshed on the Storage Analysis dashboard.
dash_storage_tblspcov-ervw_timeout	300 seconds	Specifies the number of seconds after which the Tablespace Overview pie chart is refreshed on the Storage Analysis dashboard.
dash_sys_waits_nowaits_out	300 seconds	Specifies the number of seconds after which the System Waits By Number Of Waits pie chart is refreshed on the System Waits Analysis dashboard.
dash_sys_waits_time-wait_timeout	300 seconds	Specifies the number of seconds after which the System Waits By Time Waited pie chart is refreshed on the System Waits Analysis dashboard.
dash_sys_waits_wait-dtl_timeout	300 seconds	Specifies the number of seconds after which the System Waits Details table is refreshed on the System Waits Analysis dashboard.
deleted_charts_reten-tion_time	7 days	Specifies the number of days that a custom chart (displayed on a user-defined dashboard) is stored.

continues on next page

Table 2 – continued from previous page

Parameter name	Value and unit	Description
deleted_objects_data_retention_time	3 days	Specifies the number of days after which the probe data of any deleted object, for example, server or agents or Barman server, is deleted from the pemhistory and pemdata schema. It deletes the job in case you don't want to delete the data of those obsolete objects.
deleted_probes_retention_time	7 days	Specifies the number of days that a custom probe (displayed on a user-defined dashboard) is stored.
download_chart_format	jpeg	Specifies the format in which a downloaded chart is stored. Can be jpeg or png.
flapping_detection_state_change	3	Specifies the number of state changes detected within a specified interval to define a given alert as flapping. Flapping starts when more than *N* state changes have occurred over [*N* + 1 *(min(probe_interval) * 2)] minutes and the fine state is not None. The default value of *N* is 2 or 3, and min(probe_interval) is the smallest interval for all the probes used by the alert. Flapping ends when ZERO state changes have occurred over [2 *N* * min(probe_interval)] minutes.
job_notification_error_limit	1000	Specifies the character limit for the error output included in job failure notifications.
job_notification_webhook_endpoint		Specifies the name of the webhook endpoint to be used for job failure notifications. When a job fails, a notification is sent to the configured webhook.
job_retention_time	30 days	Specifies the number of days that nonrecurring scheduled tasks and their associated jobs are retained.
long_running_transaction_minutes	5 minutes	Specifies the number of minutes a query executes before being considered long running.
nagios_cmd_file_name	file_name	Specifies nagios command file to which passive service check results are sent.
nagios_enabled	t	Specifies whether alert notification are submitted to nagios.
nagios_medium_alert_as_critical	f	Specifies whether a medium level PEM alert is considered critical in nagios.
nagios_spool_retention_time	7 days	Specifies the number of days to retain nagios messages in the spool table before they are discarded.
probe_log_retention_time	30 days	Specifies the number of days that probe log records are retained.
reminder_notification_interval	24 hours	Specifies the number of hours after which a reminder email is sent in case an alert wasn't cleared.
send_alert_reminder_mail	true	Specifies whether the PEM server should send repeated reminder emails for active alerts. When set to *false*, users will only receive the initial alert notification.
server_log_retention_time	30 days	Specifies the number of days that the server log is retained on the PEM server.
show_data_tab_on_graph	false	If *true*, a **Data** tab is added to each graph. Select the **Data** tab to review the data that's plotted on the graph.

continues on next page

Table 2 – continued from previous page

Parameter name	Value and unit	Description
smtp_authentication	false	Specifies whether to enable/disable authentication over SMTP.
smtp_enabled	true	Specifies whether to enable/disable sending of emails.
smtp_encryption	false	Specifies whether to send SMTP email using an encrypted connection.
smtp_password		Specifies the password to use to connect to the SMTP server.
smtp_port	25	Specifies the SMTP server port to use for sending email.
smtp_server	127.0.0.1	Specifies the SMTP server host address to use for sending email.
smtp_spool_retention_time	7 days	Specifies the number of days to retain sent email messages in the spool table before they are discarded.
smtp_username		Specifies the username to used to connect to an SMTP server.
snmp_authentication_password		Specifies the authentication password associated with security name mentioned in snmp_security_name. Used only for SNMPv3.
snmp_authentication_protocol	NONE	Specifies the authentication type for SNMP traps. Its possible values are NONE, HMACMD5, and HMACSHA. Used only with SNMPv3.
snmp_community	public	Specifies the SNMP community used when sending traps. Used only with SNMPv1 and SNMPv2.
snmp_context_engine_id		Specifies the context engine id, the identifier for MIB objects when sending SNMP traps. If not specified, snmp_security_engine_id is used. Used only with SNMPv3.
snmp_context_name		Specifies the context name, the identifier for MIB objects when sending SNMP traps. Used only with SNMPv3.
snmp_enabled	true	Specifies whether to enable/disable sending SNMP traps.
snmp_port	162	Specifies the SNMP server port to use for sending SNMP traps.
snmp_privacy_password		Specifies the privacy password associated with security name mentioned in snmp_security_name. Used only for SNMPv3.
snmp_privacy_protocol	NONE	Specifies the privacy protocol for SNMP traps. Its possible values are NONE, DES, AES128, IDEA, AES192, or AES256. Used only with SNMPv3.
snmp_security_engine_id		Specifies the engine id of the SNMP agent on the SNMP server. Used only with SNMPv3.

continues on next page

Table 2 – continued from previous page

Parameter name	Value and unit	Description
snmp_security_level	NOAUTH_NOPRIV	Specifies security level. Its possible values are: - AUTH_NOPRIV - Authentication, No Privacy. Then NONE option is possible for privacy protocol only. AUTH_PRIV - Authentication, Privacy. Then NONE option selection shouldn't be allowed for authentication and privacy protocol. NOAUTH_NOPRIV - no Authentication, no Privacy. Then only NONE option is possible for authentication and privacy protocol. Used only with SNMPv3.
snmp_security_name		Specifies the user name or security name for sending SNMP traps. Used only with SNMPv3.
snmp_server	127.0.0.1	Specifies the SNMP server host address to use for sending SNMP traps.
snmp_spool_retention_time	7 days	Specifies the number of days to retain sent traps in the spool table before they're discarded.
timezone_for_system_jobs	current server timezone	Specifies the time zone to be used when scheduling the next run time for system jobs.
webclient_help_pg	EDB hosted documentation	Specifies the location of the online PostgreSQL core documentation.

CHAPTER 19

Managing Postgres servers

19.1 Starting and stopping monitored database servers

PEM lets you start up and shut down managed server instances with the PEM client. To configure a server to allow PEM to manage the service, complete the Server Registration dialog box. Use the dialog box to register the database server with a PEM agent, which is the local agent installed on the same host as the database server. You can also:

- Specify the **Store on PEM Server** option on the Properties dialog box.
- Specify the name of a service script in the **Service ID** field on the **Advanced** tab:
- For EDB Postgres Advanced Server, the service name is `edb-as-<x>` or `ppas-<x>` .
- For PostgreSQL, the service name is `postgresql-<x>` .

Where `<x>` indicates the server version number.

After connecting to the server, you can start or stop the server by selecting the server name in the tree and selecting **Tools > Queue Server Startup** or **Tools > Queue Server Shutdown**.

19.2 Modifying the pg_hba.conf file

Entries in the `pg_hba.conf` file control network authentication and authorization. The `pg_hba.conf` file on the PEM server host must allow connections between the PEM server and PEM-HTTPD, the PEM agent, and the monitored servers.

The PEM server installation process prompts you for the IP address and connection information for hosts for PEM to monitored. This information is added to the top of the `pg_hba.conf` file of the PEM backing database.

You might also need to manually modify the `pg_hba.conf` file to allow connections between the PEM server and other components. For example, if your PEM-HTTPD installation isn't on the same host as the PEM server, you must modify the `pg_hba.conf` file on the PEM server host to allow PEM-HTTPD to connect to the server.

By default, the `pg_hba.conf` file resides in the data directory, under your Postgres installation. For example, on an EDB Postgres Advanced Server 10 host, the default location of `pg_hba.conf` is:

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

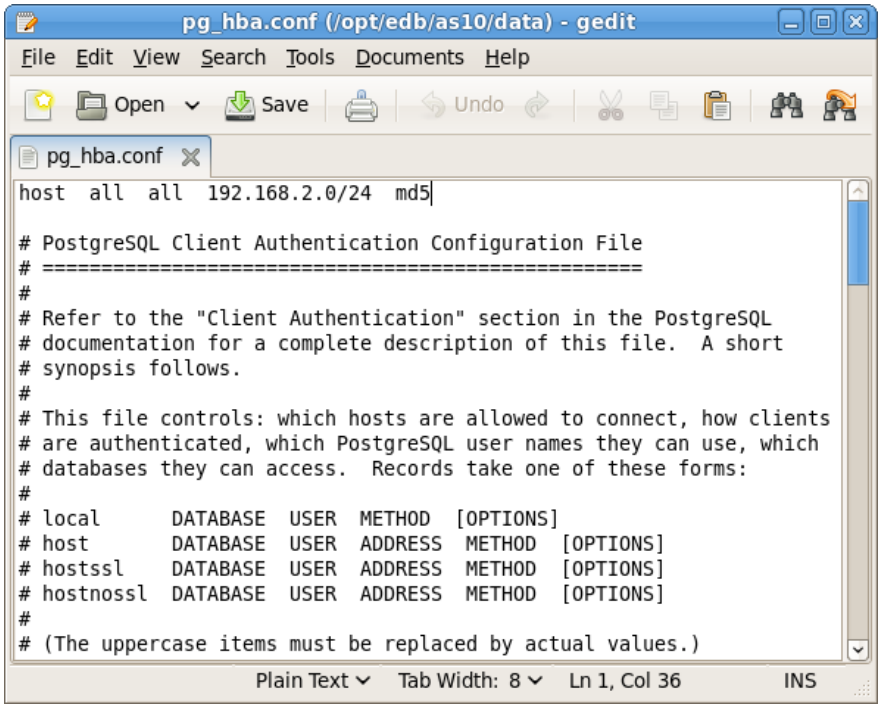


Fig. 1: PEM entries in the pg_hba.conf file

You can modify the `pg_hba.conf` file with your editor of choice. After modifying the file, reload the server for changes to take effect.

The following example shows a `pg_hba.conf` entry that allows an md5 password authenticated connection from a user named postgres to the `postgres` database on the host where `pg_hba.conf` resides. The connection is coming from an IP address of 192.168.10.102:

```
#  TYPE     DATABASE        USER        CIDR-ADDRESS          METHOD

#  IPv4 local connections:

 host     postgres       postgres    192.168.10.102/32       md5
```

You can specify the address of a network host or a network address range. For example, if you want to allow connections from servers with the addresses 192.168.10.23, 192.168.10.76, and 192.168.10.184, enter a CIDR-ADDRESS of `192.168.10.0/24` to allow connections from all of the hosts in that network:

```
#  TYPE     DATABASE        USER        CIDR-ADDRESS       METHOD

#  IPv4 local connections:

  host      postgres         all       192.168.10.0/24       md5
```

For more information about formatting a `pg_hba.conf` file entry, see the PostgreSQL core documentation .

Before you can connect to a Postgres server with PEM, you must ensure that the `pg_hba.conf` file on both servers allows the connection. If you receive an error when connecting to the database server, modify the `pg_hba.conf` file, adding an entry that allows the connection.

19.3 Creating and maintaining databases and objects

Each instance of a Postgres server manages one or more databases. Each user must provide authentication information to connect to the database before accessing the information it contains. The PEM client lets you create and manage databases and the objects that comprise a database, such as tables, indexes, and stored procedures.

To create a database in PEM, right-click any managed server's **Databases** node and select **Create > Database**. After defining a database, you can create objects in the new database.

For example, to create a table, right-click a **Tables** node and select **Create > Table**. Specify the attributes of the table in the New Table dialog box.

PEM provides similar dialog boxes for creating and managing other database objects such as:

- Tables
- Indexes
- Stored procedures
- Functions
- Triggers
- Views
- Constraints

Each object type is listed in the tree. Right-click the node that corresponds to an object type to access the **Create** menu and create an object. To perform administrative tasks for the selected object, select **Properties** from the context menu of a named node.

19.4 Template Linux service script

A service script for the database server allows the PEM server to start, stop, or restart the database server. Doing so might be necessary when performing configuration management, certificate management, and other administrative tasks. Service scripts are platform specific.

The Postgres server on which the PEM server resides must contain a service script. Postgres installers in Windows generated by EDB create a service script for you. If you're using a Postgres server from another source like native packages, you must provide a service script.

> **Note**
>
> On Rocky Linux or RHEL 8.x, the service script resides in */usr/lib/systemd/system* .

For information about customizing a Postgres service, see the PostgreSQL documentation .

CHAPTER 20

Managing a PEM agent

Use the Postgres Enterprise Manager browser-based console to manage PEM agents.

The PEM agent executes tasks and reports statistics from the agent host and the monitored Postgres instances to the PEM server. A single PEM agent can monitor multiple installed instances of Postgres that reside on the same host where the agent is installed or on multiple remote hosts. It's also responsible for executing other tasks that you might schedule, such as server shutdowns, SQL Profiler traces, and custom jobs.

To manage the PEM agents, you can:

- *Managing job notifications*
- *Managing scheduled jobs*
- *Modifying agent configuration*
- *Reviewing or modifying agent properties*
- *Setting agent privileges*

20.1 Managing job notifications

In the PEM console, you can configure the settings for sending the SMTP trap on success or failure of a system-generated job. These jobs are listed under scheduled tasks or are a custom agent job. You can configure these email notification settings at the job, agent, or PEM server level (the default) to send email notifications to the specified user group. The order of precedence is job, agent, and PEM server.

20.1.1 Configuring job notifications at job level

You can configure email notification settings at the job level only for a custom agent job.

- For a new agent job, you can configure the email notification settings in the **Notification** tab of the Create Agent Job wizard.
- For an existing custom job, you can edit the properties of the job and configure the notification settings.

On the **Notifications** tab:

- In the **Send the notifications** field, specify when you want to send the email notifications.
- In the **Email group** field, specify the email group to send the email notification to.

20.1.2 Configuring job notifications at agent level

In the tree view, right-click the agent and select **Properties**. In the Properties dialog box, select the **Job Notifications** tab.

- Set **Override default configuration?** to **Yes** if you want the agent level job notification settings to override the default job notification settings. The rest of the settings then become enabled.
- Set **Email on job completion?** to **Yes** to send a notification when the job completes successfully.
- Set **Email on a job failure?** to **Yes** to send the job notification when the job fails.
- In the **Email group** field, specify the email group to send the job notification to.

20.1.3 Configuring job notifications at server level

To configure your job notifications at the PEM server level, select **Management > Server Configuration**. In the Server Configurations dialog box:

- Select **job_failure_notification** to send an email notification after each job failure.
- Use the **job_notification_email_group** parameter to specify the email group to send the email notification to.
- Use **job_retention_time parameter** to specify the number of days to retain nonrecurring scheduled tasks in the system.
- Select **job_status_change_notification** to send an email notification after each status change of failure, success, or interrupted.

20.2 Managing scheduled jobs

You can create a PEM scheduled job to perform a set of steps you define in a specified sequence. These steps can contain SQL code or a batch/shell script that you can run on a server that's bound with the agent. For example, you can create a job for taking a backup of a database server and schedule when it runs each month.

To create or manage a PEM scheduled job, use the PEM tree to browse to the PEM agent whose job you want to create. To add a job, right-click the **Jobs** node and select **Create Job**.

Use the tabs on the Create - Agent Job dialog box to define the steps and schedule that make up a PEM scheduled job.

20.2.1 Provide general information about the job

Use the **General** tab to provide general information about a job:

- Provide a name for the job in the **Name** field.
- Set **Enabled** to **Yes** to enable a job.
- Enter notes about the job in the **Comment** field.

20.2.2 Define the job steps

Use the **Steps** tab to define and manage the steps that the job performs. Select **Add** (+) to add a step. Then, select the compose icon, to the left of the header, to open the Step Definition dialog box where you define the step:

- Enter the name for the step in the **Name** field. Steps are performed in alphanumeric order by name.

- Set **Enabled** to **True** to include the step when executing the job.
- Set the **Kind** switch to set the job step to invoke SQL code or a batch script. Then use the **Code** tab to enter the SQL code or the batch script for the step.
- Use the **On error** list to specify the pemAgent behavior if it encounters an error while executing the step:
- **Fail** _ Stop the job if you encounter an error while processing this step.
- **Success** _ Mark the step as completing successfully and continue.
- **Ignore** _ Ignore the error and continue.
- If you selected **SQL** as your input for the **Kind** switch:
- In the **Server** field, enter the server that's bound with the agent for which you're creating the PEM scheduled job.
- In the **Database** field, enter the database associated with the server you selected.
- In the **Comment** field, enter a comment about the step.
- Use the context-sensitive field on the Step Definition dialog box **Code** tab to provide the SQL code or batch script to execute during the step:
- If the step invokes SQL code, provide one or more SQL statements in the **SQL query** field.
- If the step invokes a batch script, provide the script in the **Code** field. If you're running on a Windows server, use standard batch file syntax. On a Linux server, you can use any shell script, provided that you specify a suitable interpreter on the first line (such as `#!/bin/sh`). Along with the defined inline code, you can also provide the path of any batch script, shell script, or SQL file on the filesystem.

```
-    To invoke a script on a Linux system, you must modify the entry for the `batch_
↪script_user`  parameter in the `agent.cfg`  file and specify the user who runs the␣
↪script. You can specify either a nonroot user or root for this parameter. If you don
↪'t specify a user or the specified user doesn't exist, then the script doesn't␣
↪execute. Restart the agent after modifying the file.

-    To invoke a script on a Windows system, set the registry entry for␣
↪`AllowBatchJobSteps`  to `true`  and restart the PEM agent. PEM registry entries␣
↪are located in `HKEY_LOCAL_MACHINE\Software\EnterpriseDB\PEM\agent` .
```

After providing all the information required by the step, select **Save**.

20.2.3 Define the job schedule

Select **Add** (+) to add each step, or select the **Schedules** tab to define the job schedule.

On the **Schedules** tab, select **Add** (+) to add a schedule for the job. Then select the compose icon located to the left of the header to open the Schedule Definition dialog box.

Use the **Schedules definition** tab to specify the days and times for the job to execute:

- Enter a name for the schedule in the **Name** field.
- Set **Enabled** to **Yes** to use the schedule (**Yes**).
- Use the calendar selectors to specify the start and end date and time for the schedule.
- In the **Comment** field, enter a comment about the schedule.

Select the **Repeat** tab to define the days when the schedule executes in a cron-style format. The job executes on each date or time element selected on the **Repeat** tab.

In each field, select a value to add it to the list of selected values for the field.

- Use the fields in the **Days** box to specify the days when the job executes:
- Use the **Week Days** field to select the days when the job executes.
- Use the **Month Days** field to select the numeric days when the job executes. Select **Last Day** to perform the job on the last day of the month, regardless of the date.
- Use the **Months** field to select the months when the job executes.
- Use the fields in the **Times** box to specify the times when the job executes in hours and minutes.

Select the **Exceptions** tab to specify any days when you don't want the schedule to execute. For example, you might not want to run jobs on holidays.

Select **Add** (+) to add a row to the exception table. Then:

- In the **Date** column, open a calendar selector and select a date when you don't want the job to execute. Specify ****<Any>**** to indicate that you don't want the job to execute on any day at the time selected.
- In the **Time** column, open a time selector and specify a time when you don't want the job to execute. Specify ****<Any>**** to indicate that you don't want the job to execute at any time on the day selected.

20.2.4 Configure the email notifications

Select the **Notifications** tab to configure the email notification settings on job level:

- In the **Send the notifications** field, specify when you want to send the email notifications.
- In the **Email group** field, specify the email group to send the email notification to.

When you finish defining the schedule, you can use the **SQL** tab to review the code that creates or modifies your job.

Select **Save** to save the job definition.

After you save a job, the job is listed under the **Jobs** node of the PEM tree of the server on which it was defined. The **Properties** tab in the PEM console displays a high-level overview of the selected job, and the **Statistics** tab shows the details of each run of the job.

To modify an existing job or to review detailed information about a job, right-click a job name, and select **Properties** from the context menu.

20.3 Modifying agent configuration

You can configure parameters to control PEM agent behavior. On Linux, these parameters are stored in a configuration file. On Windows they're stored as registry keys.

Most agent configuration is managed automatically. We recommend that you don't manually modify any of these configuration parameters unless specifically told to.

> **Note**
>
> If you add, remove, or modify any of the parameters, restart the agent to apply the changes.

20.3.1 Contents of the configuration file

On Linux systems, PEM configuration parameters are stored in the `agent.cfg` file, located in `/usr/edb/pem/agent/etc`.

Parameter name	Description	Default value
pem_host	The IP address or hostname of the PEM server. This can be a comma-separated list of IPs or hostnames if you're using an *Query information* . See the libpq documentation for more details on the syntax.	127.0.0.1.
pem_port	The database server port to which the agent connects to communicate with the PEM server. This can be a comma-separated list of ports if you're using an *Query information* . See the libpq documentation for more details on the syntax.	Port 5432.
agent_conn_string	A connection string used by the agent for connections to the PEM backend. See *Using connection strings* .	
pem_agent	A unique identifier assigned to the PEM agent.	The first agent is '1', the second agent is '2', and so on.
agent_ssl_key	The complete path to the PEM agent's key file.	/root/.pem/agent.key
agent_ssl_crt	The complete path to the PEM agent's certificate file.	/root/.pem/agent.crt
agent_flag_dir	Used for HA support. Specifies the directory path checked for requests to take over monitoring another server. Requests are made in the form of a file in the specified flag directory.	Not set by default.
log_level	The type of event to write to the PEM log files: *debug2*, *debug1*, *info*, *warning*, or *error*. These are in descending order of logging verbosity; *debug2* logs everything possible, and *error* logs only errors.	warning
log_location	The location of the PEM worker log file.	/var/log/pem/worker.log
agent_log_location	The location of the PEM agent log file.	/var/log/pem/agent.log
long_wait	The maximum length of time (in seconds) for the PEM agent to wait before attempting to connect to the PEM server if an initial connection attempt fails.	30 seconds
short_wait	The minimum length of time (in seconds) for the PEM agent to wait before checking which probes are next in the queue waiting to run.	10 seconds
alert_threads	The number of alert threads to be spawned by the agent. For more information, see *About alert threads* .	Set to 1 for the agent that resides on the host of the PEM server, 0 for all other agents.
enable_smtp	When set to true, this agent will attempt to send email notifications as configured in the PEM web application.	true for PEM server host, false for all others.
enable_snmp	When set to true, this agent will attempt to send SNMP notifications as configured in the PEM web application.	true for PEM server host, false for all others.
enable_nagios	When set to true, Nagios alerting is enabled.	true for PEM server host, false for all others.
enable_webhook	When set to true, Webhook alerting is enabled.	true for PEM server host, false for all others.
max_webhook_retries	The maximum number of times the PEM agent retries calls to webhooks on failure.	Default 3.
connect_timeout	The maximum time in seconds (a decimal integer string) for the agent to wait for a connection.	Not set by default. Set to 0 to indicate for the agent to wait indefinitely.

continues on next page

Table 1 – continued from previous page

Parameter name	Description	Default value
allow_server_restart	If set to TRUE, the agent can restart the database server that it monitors. Some PEM features might be enabled/disabled, depending on the value of this parameter.	False
max_connections	The maximum number of probe connections used by the connection throttler.	0 (an unlimited number)
connection_lifetime	Use ConnectionLifetime (or connection_lifetime) to specify the minimum number of seconds an open but idle connection is retained. This parameter is ignored if the value specified in MaxConnections is reached and a new connection to a different database is required to satisfy a waiting request.	By default, set to 0 (a connection is dropped when the connection is idle after the agent's processing loop).
allow_batch_probes	If set to TRUE, the user can create batch probes using the custom probes feature.	false
heartbeat_connection	When set to TRUE, a dedicated connection is used for sending the heartbeats. If set to TRUE, the *max_connections* parameter must be set to greater than 1.	false
batch_script_dir	The path where script file (for alerting) is stored.	/tmp
connection_custom_setup	Used to provide SQL code to invoke when a new connection with a monitored server is made.	Not set by default.
ca_file	The path to a CA certificate to use instead of the platform default for verifying HTTPS server certificates. This will be used for webhook and PG Backup API endpoints, used for Barman monitoring, unless overridden by a more specific **_ssl_ca_crt* setting (see below).	Not set by default.
batch_script_user	The name of the user to use for executing the batch/shell scripts.	None
alert_reconnect_delay	Length of time in seconds for alert threads to wait after connecting to the PEM server before starting to evaluate alerts. This helps prevent false alarms caused by alerts being evaluated before the server has had the chance to receive heartbeats from the agents.	30
Webhook parameters	You can specify the following options multiple times. Each time, precede the option with a header of the form *[WEBHOOK/<name>]*, where *<name>* is the name of a previously created webhook. These settings are inserted when webhooks are added using the pemworker utility. If you add a webhook from the PEM web application, you must add the required settings manually.	
webhook_ssl_crt	The complete path to a client certificate to use when connecting to the webhook endpoint.	
webhook_ssl_key	The complete path to the key associated with *webhook_ssl_crt*.	
webhook_ssl_crl	The complete path of the CRL file to use when validating the webhook endpoint's server certificate.	
webhook_ssl_ca_crt	The complete path to a CA certificate file to use when validating the webhook endpoint's server certificate.	

continues on next page

Table 1 – continued from previous page

Parameter name	Description	Default value
allow_insecure_webhooks	When set to true, prevents the agent from verifying the endpoint server certificate. This option can be used to allow PEM to call plain HTTP endpoints as well as untrusted HTTPS endpoint. Use with caution.	false
Barman parameters	You can specify the following options multiple times. Each time, precede the option with a header of the form *[BARMAN/<n>]*, where *<n>* is the ID of a previously registered Barman server. These settings are inserted when Barman is registered added using the pemworker utility. If you register Barman from the PEM web application, you must add the required settings manually.	
ssl_crt	The complete path to a client certificate to use when connecting to the PG Backup API endpoint.	
ssl_key	The complete path to the key associated with *ssl_crt*.	
ssl_ca_crt	The complete path to a CA certificate file to use when validating the PG Backup API endpoint's server certificate.	

20.3.2 Contents of the registry

On Windows systems, PEM registry entries are located in:

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

The registry contains the entries shown in the table.

Parameter name	Description	Default value
PEM_HOST	The IP address or hostname of the PEM server.	127.0.0.1.
PEM_PORT	The database server port to which the agent connects to communicate with the PEM server.	Port 5432.
AGENT_CONN_STRIN	A connection string used by the agent for connections to the PEM backend. See *Using connection strings* .	
AgentID	A unique identifier assigned to the PEM agent.	The first agent is '1', the second agent is '2', and so on.
AgentKeyPath	The complete path to the PEM agent's key file.	%APPDATA%\Roaming\pem\agent.key.
AgentCrtPath	The complete path to the PEM agent's certificate file.	%APPDATA%\Roaming\pem\agent.crt
AgentFlagDir	Used for HA support. Specifies the directory path checked for requests to take over monitoring another server. Requests are made in the form of a file in the specified flag directory.	Not set by default.
LogLevel	Specifies the type of event to write to the PEM log files. These are in descending order of logging verbosity; *debug2* logs everything possible, and *error* only logs errors.	warning
LongWait	The maximum length of time (in seconds) that the PEM agent waits before attempting to connect to the PEM server if an initial connection attempt fails.	30 seconds

continues on next page

Table 2 – continued from previous page

Parameter name	Description	Default value
shortWait	The minimum length of time in seconds that the PEM agent waits before checking which probes are next in the queue (waiting to run).	10 seconds
AlertThreads	The number of alert threads for the agent to spawn. For more information, see *About alert threads* .	Set to 1 for the agent that resides on the host of the PEM server, 0 for all other agents.
EnableSMTP	When set to true, the SMTP email feature is enabled.	true for PEM server host, false for all others.
EnableSNMP	When set to true, the SNMP trap feature is enabled.	true for PEM server host, false for all others.
EnableWebhook	When set to true, Webhook alerting is enabled.	true for PEM server host, false for all others.
MaxWebhookRetries	Sets the maximum number of times the PEM agent retries calls to webhooks on failure.	Default 3.
ConnectTimeout	The maximum time in seconds (a decimal integer string) that the agent waits for a connection.	Not set by default. If set to 0, the agent waits indefinitely.
AllowServerRestart	If set to TRUE, the agent can restart the database server that it monitors. Some PEM features might be enabled/disabled, depending on the value of this parameter.	true
MaxConnections	The maximum number of probe connections used by the connection throttler.	0 (an unlimited number)
ConnectionLifetime	Use ConnectionLifetime (or connection_lifetime) to specify the minimum number of seconds an open but idle connection is retained. This parameter is ignored if the value specified in MaxConnections is reached and a new connection to a different database is required to satisfy a waiting request.	By default, set to 0 (a connection is dropped when the connection is idle after the agent's processing loop).
AllowBatchProbes	If set to TRUE, the user can create batch probes using the custom probes feature.	false
HeartbeatConnection	When set to TRUE, a dedicated connection is used for sending the heartbeats.	false
BatchScriptDir	The path to store the script file for alerting.	/tmp
ConnectionCustomSetup	Used to provide SQL code to invoke when a new connection with a monitored server is made.	Not set by default.
AllowBatchJobSteps	If set to true, the batch/shell scripts are executed. Scripts are executed by the user account under which the PEM agent is running.	None
AlertReconnectDelay	A length of time in seconds. Alert threads will wait for this time after connecting to the PEM Server before starting to evaluate alerts. This helps prevent false alarms caused by alerts being evaluated before the server has had the chance to receive heartbeats from the agents.	30

continues on next page

Table 2 – continued from previous page

Parameter name	Description	Default value
Webhook parameters	You can specify the following options multiple times. They must be placed in a subkey *SoftwareEnterpriseDBPEMagentWEBHOOK<name>*, where *<name>* is the name of a previously created webhook. These settings are automatically added when webhooks are created. We don't recommended adding them manually.	
WebhookSSLCrt	The complete path to a client certificate that will be used when connecting to the webhook endpoint.	
WebhookSSLKey	The complete path to the key associated with *webhook_ssl_crt*.	
WebhookSSLCrl	The complete path of the CRL file to use when validating the webhook endpoint's server certificate.	
WebhookSSLCaCrt	The complete path to a CA certificate file to to use when validating the webhook endpoint's server certificate.	
AllowInsecureWebhooks	When set to true, prevents the agent from verifying the endpoint server certificate. This option can be used to allow PEM to call plain HTTP endpoints as well as untrusted HTTPS endpoint. Use with caution.	false
Barman parameters	You can specify the following options multiple times. They must be placed in a subkey *SoftwareEnterpriseDBPEMagentBARMAN<n>*, where *<n>* is the ID of a previously registered Barman server. These settings are automatically added when a Barman server is registered. We don't recommend adding them manually.	
SSLCrt	The complete path to a client certificate that will be used when connecting to the PG Backup API endpoint.	
SSLKey	The complete path to the key associated with *ssl_crt*.	
SSLCaCrt	The complete path to a CA certificate file to use when validating the PG Backup API endpoint's server certificate.	

20.3.3 About alert threads

The number of alert threads spawned by an agent is determined by the `alert_threads` or `AlertThreads` parameter. In general, we recommend setting this parameter to 1 on the agent that resides on the PEM server and 0 for all other agents. However, on PEM server instances with very large numbers of alerts (caused by many monitored servers, many enabled alerts, or high alert frequency), you may need to increase this parameter if alerts aren't being evaluated at the configured frequency. In this situation, we recommend setting this parameter to around 8 on the agent that resides on the PEM server and 0 for all other agents and tuning up or down accordingly.

When tuning this parameter, any agent can process any alert, so in general you don't need a non-zero number of alert threads on more than one agent. The capacity of the PEM instance to process alerts is determined by the total number of alert thread across all agents. Increasing the number of threads on a specific agent doesn't give any additional performance for alerts pertaining to servers monitored by that agent.

Each alert thread opens a connection to the PEM server backend, so allocating more threads than necessary results in additional memory and CPU usage on the PEM server.

20.4 Reviewing or modifying agent properties

Use the PEM Agent Properties dialog box to get information about a selected PEM agent. Right-click an agent name in the PEM client tree and select **Properties** from the context menu. You can then review or modify information about the PEM agent:

- Use the **Description** field to view or edit the description of the PEM agent, which appears in the tree of the PEM client.
- You can use groups to organize your servers and agents in the PEM client tree. Use the **Group** list to select the group in which the agent is displayed.
- Use the **Team** field to specify the name of the group role that can access servers monitored by the agent. The servers monitored by this agent are visible in the PEM client tree to connected team members. This is a convenience feature. The **Team** field doesn't provide true isolation. Don't use it for security purposes.
- Use the **Heartbeat Tolerance** fields to view or edit how late a heartbeat can be before it is considered missed. Use the selectors next to the **Minutes** or **Seconds** fields to modify the value.
- Use the **Job Notifications** tab to configure the email notification settings on agent level. See *Managing job notifications* .
- The **Agent Configurations** tab displays all the current configurations and capabilities of an agent.

```
-   The **Parameter** column displays a list of parameters. -   The **Value** column␣
↪displays the current value of the corresponding parameter. -   The **Category**␣
↪column displays the category (configuration or capability) of the corresponding␣
↪parameter.
```

20.5 Setting agent privileges

By default, the PEM agent is installed with root privileges for the operating system host and superuser privileges for the database server. These privileges allow the PEM agent to invoke unrestricted probes on the monitored host and database server about system usage, retrieving and returning the information to the PEM server.

20.5.1 Root user versus non-root user

For complete functionality, run the PEM agent as root. The table gives a high-level summary of the effects of limiting privileges.

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

continues on next page

Table 3 – continued from previous page

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 *Functionality affected by limiting operating system privileges* .	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 *Functionality affected by limiting operating system privileges* .	Some of the PEM probes don't return information, and some of the functionality might be affected.
Scheduled Tasks	Limited. See the *Functionality affected by limiting operating system privileges* .	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.

20.5.2 Functionality affected by limiting operating system privileges

If you run the PEM agent as a non-root user, the level of functionality depends on the permissions the agent user has. TSome operations are impacted by OS user permissions and certain permissions are required for normal operation.

Probes

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.

continues on next page

Table 4 – continued from previous page

Probe	Operating system	PEM functionality affected
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.

Restarting services

Audit Log Manager and Server Log Manager require the PEM agent user to restart the Postgres service for changes to take effect and the features to work. The agent user needs privileges to restart services. Typically, this requires root access.

Batch/shell tasks

On Windows, the PEM agent runs batch tasks only if the agent user has administrative privileges.

On Linux, the PEM agent can run shell tasks only if the agent user can become the `batch_script_user` specified in `agent.cfg` . This is always true for the root user and the `batch_script_user` .

20.5.3 Functionality affected by limiting database privileges

If the PEM agent connects to the monitored database using a non-superuser account, the available functionality is limited based on the privileges granted to that user.

The PEM agent reads data from the pg_catalog schema for most SQL-based probes. In general, assigning the pg_monitor role to the agent user is sufficient. However, certain catalog functions and probes may require privileges beyond pg_monitor.

Also, the agent user must be able to connect to all target databases where probes need to run. If the agent can't connect to a database, no database-level probes will be executed on that instance. Only server-level metrics—such as those collected from `pg_stat_database` —are available in such cases.

The table lists probes that require permissions in addition to pg_monitor on the Linux and Windows operating systems.

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.

continues on next page

Table 5 – continued from previous page

Probe	Additional permissions required
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.

20.5.4 Error handling

If the probe is querying the operating system without enough privileges, the probe might return a `permission denied` error. If the probe is querying the database without enough privileges, the probe might return a `permission denied` error or display the returned data in a PEM chart or graph as an empty value.

When a probe fails, an entry is written to the log file that contains the name of the probe, the reason the probe failed, and a hint to help you resolve the problem.

You can view probe-related errors that occurred on the server in the Probe Log dashboard or review error messages in the PEM worker log files. On Linux, the default location of the log file is:

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

On Windows, log information is available on the Event Viewer.

CHAPTER 21

Managing SSL certificates

PEM uses SSL certificates:

- To secure requests to the *Web server certificates* , which provides the user interface and REST API.
- To secure and authenticate the *PEM backend database server and agent connection certificates* .

21.1 Web server certificates

PEM generates an SSL certificate and key file for the web server during initial configuration. Because the certificate is self-signed, a warning states that the site is insecure when users open the PEM web application URL in a browser.

To increase security and remove this warning, you can replace the self-signed SSL certificate with a certificate signed by a trusted certificate authority.

21.1.1 Replace the web server certificates (NGINX)

To use your own SSL certificate with NGINX, update the configuration file:

- On RHEL-like systems: `/etc/nginx/conf.d/edb-pem.conf`
- On Debian-like systems: `/etc/nginx/sites-available/edb-pem.conf`

Change the server name and file paths in the configuration file to match your certificate files:

```
server {
   # lines omitted here
   server_name  <yourdomain.com>;
   # lines omitted here
}

server {
   # lines omitted here
   server_name  <yourdomain.com>;
```

(continues on next page)

(continued from previous page)

```
    ssl_certificate /path/to/your_domain_name.crt
    ssl_certificate_key /path/to/your_private.key
    # lines omitted here
}
```

21.1.2 Replace the web server certificates (Apache HTTPD)

To use your own SSL certificate with Apache HTTPD, update the configuration file `edb-ssl-pem.conf` . Change the server name and file paths in the configuration file to match your certificate files.

```
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 .

21.2 PEM backend database server and agent connection certificates

PEM implements secured SSL/TLS connections between PEM agents and the backend database. Each agent has an SSL certificate that's used both to encrypt its communication with the server and to authenticate with the server in place of a password.

PEM uses the sslutils extension to allow the PEM server to generate and sign SSL certificates and keys. When a new agent is registered, the PEM server issues it a certificate. Certificates issued by the PEM server are signed by the PEM server, meaning the PEM server is acting as a certificate authority (CA).

If this approach isn't suitable, you can use SSL certificates and keys generated outside of PEM and signed by a trusted CA. For more information, see *Use certificates and keys signed by trusted CA* .

21.2.1 Certificates and key files on the PEM server

During initial configuration of the PEM server, the following files are generated in the Postgres data directory of the PEM server:

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

The `ca_certificate.crt` and `ca_key.key` files are used by the PEM server to sign certificates generated for agents during agent registration. They're also used to sign `server.crt` . Unless replaced manually, the 'ca_certificate.crt' file is a self-signed certificate because it's acting as the root CA.

The `root.crt` file is a copy of the `ca_certificate.crt` file. The `ssl_ca_file` parameter in the `postgresql.conf` file points to this file.

The `root.crl` is the certificate revocation list (CRL) of certificates revoked by the issuing CA before their actual or assigned expiration date. The `ssl_crl_file` parameter in the `postgresql.conf` file points to this file.

The `server.crt` file is the signed certificate for the PEM server, and the `server.key` file is the private key to the certificate. The `ssl_cert_file` parameter in the `postgresql.conf` file points to this file.

These files are automatically renewed when they near their expiry date. See *PEM certificate renewal* .

21.2.2 Certificates and key files for PEM agents

Each agent's SSL certificate and keys are generated during *Registering a PEM agent* . The PEM agent connects to the PEM backend database server using the libpq interface, acting as a client of the backend database server. The PEM agent connects to the server using the `cert` auth method and with ssl enabled. This means that the connection is encrypted using the agent's key and authenticated using the agent's certificate instead of, for example, a password.

Each agent has a unique identifier, and the agent certificates and keys have the corresponding identifier.

If required, you can use the same certificate for all agents rather than one certificate per agent. For more information, see *Generate a common agent certificate and key pair* .

For more information on using the SSL certificates to connect in Postgres, see Securing TCP/IP connections with SSL in the Postgres documentation.

21.2.3 PEM certificate renewal

SSL certificates have an expiry date. If you're using certificates and keys generated by PEM, PEM replaces them before they expire. The PEM agent installed with the PEM server monitors the expiration date of the `ca_certificate.crt` file. When the certificate is about to expire, PEM:

- Makes a backup of the existing certificate files.
- Creates new certificate files and appends the new CA certificate file to the `root.crt` file on the PEM server.
- Creates a job to renew the certificate file for any active agents.
- Restarts the PEM server.

Important

If you choose to provide your own certificates or use a single certificate for all agents, disable the automatic renewal job. On the PEM server, execute the following SQL:

```
UPDATE pem.job
SET jobenabled=false
WHERE jobname = Check CA certificate expiry;
```

- *Regenerating the agent SSL certificates*

21.3 Generate a common agent certificate and key pair

By creating and using a single Postgres user for all PEM agents rather than one user per agent (the default), you can use the same certificate for all agents.

Create a user, generate an agent certificate and key pair, and use them for all PEM agents.

1. Create one common agent user in the PEM backend database. Grant the pem_agent role to the user.

```
# 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. Generate an agent key:

```
# Running as root
openssl genrsa -out agent.key 4096
```

1. Generate a CSR for the agent:

```
openssl req -new -key agent.key -out agent.csr -subj /C=IN/ST=MH/L=Pune/O=PEM/CN=
↪<agent_user>
```

Where `-subj` is provided as per your requirements.

1. Use the `openssl x509` command to sign the CSR and generate an agent certificate:

```
openssl x509 -req -days 365 -in agent.csr -CA ca_certificate.crt -CAkey ca_key.key -
↪CAcreateserial -out agent.crt
```

1. Change the permissions on the `agent.crt` and `agent.key` files:

```
chmod 600 agent.crt agent.key
```

1. Use this agent certificate and key pair:
   - For registering the new PEM agent from the remote host to the PEM server.

a. Copy the agent certificate and key pair to the remote agent host and register the agent:

```
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 pem_agent_common_user \
-o agent_ssl_crt= agent.crt \
-o agent_ssl_key= agent.key
```

b. Enable and start the pemagent services:

```
systemctl enable pemagent
systemctl start pemagent
```

- To replace the agent certificate and key pair with the registered agent. a. Edit the `agent_user` , `agent_ssl_key` , and `agent_ssl_crt` parameters in the `agent.cfg` file of the agent host:

```
vi /usr/edb/pem/agent/etc/agent.cfg
# Edit the agent username
agent_user=pem_agent_common_user
# Edit the ssl parameters with new certificate and key file location
agent_ssl_key=<new_location>/agent.key
agent_ssl_crt=<new_location>/agent.crt
```

b. Restart the pemagent service:

```
systemctl restart pemagent
```

21.4 Use certificates and keys signed by trusted CA

You can replace the PEM SSL certificates and keys with certificates and keys signed by a trusted CA.

After obtaining the trusted CA certificates and keys, replace the *Replace the server SSL certificates with certificates signed by a trusted CA* and *Replace the agent SSL certificates with certificates signed by a trusted CA* certificates and keys.

21.4.1 Replace the server SSL certificates with certificates signed by a trusted CA

1. Back up the old server certificate and key files:

```
# 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. Generate a private key for the server:

```
openssl genrsa -out server.key 4096
```

1. Generate a CSR for the server:

```
openssl req -new -key server.key -out server.csr -subj /C=IN/ST=MH/L=Pune/O=EDB/CN=PEM
```

Where `-subj` is provided as per your requirements. We recommend using the hostname or domain qualified full name of the PEM server host for `CN` .

1. Obtain the CA certificate (`trusted_ca.crt`) from a trusted CA.
2. Ask your CA to sign the CSR and generate the server certificate for you.
3. Verify that the details of the new server certificate aren't tampered with and match your provided details:

```
openssl x509 -noout -text -in server.crt
```

1. Use the new certificate obtained from the CA as the `root.crt` file:

```
cp trusted_ca.crt root.crt
```

1. If the trusted CA doesn't provide CRL, disable CRL usage by the server. To disable the CRL usage, comment the `ssl_crl_file` parameter in the `postgresql.conf` file.

> **Note**
>
> If you leave a CRL from a previous CA in place and don't comment out *ssl_crl_file* , the server will start. However, authentication will fail with an SSL error message: *tlsv1 alert unknown ca* . The error doesn't specify that the CRL is the cause, so this issue can be difficult to debug if encountered out of context.

1. Copy the new `root.crt` , `server.key` , and `server.crt` files to the data directory of the backend database server:

```
cp root.crt server.key server.crt /var/lib/edb/as<x>/data
```

1. Change the owner and permissions of the new certificates and key files to the same name as the data directory:

```
cd /var/lib/edb/as<x>/data/
chown enterprisedb server.* root.crt ca_certificate.crt
chmod 600 server.* root.crt ca_certificate.crt
```

Note

Don't restart the PEM server now. If you restart the PEM server, all the registered agents will stop working.

1. Replace each PEM agent SSL certificates with the trusted CA certificates. For more information, see these *Replace the agent SSL certificates with certificates signed by a trusted CA* .
2. Restart the PEM backend database server.

21.4.2 Replace the agent SSL certificates with certificates signed by a trusted CA

Replace the agent SSL certificates only after replacing the server certificates `server.crt` and `server.key` and CA certificate `root.crt` .

1. Use psql to find all the agent identifiers (IDs) needed to replace the SSL certificates:

```
psql -U enterprisedb -d pem --no-psqlrc -t -A -c "SELECT id FROM pem.agent WHERE␣
↪active=true"
```

1. After identifying the agents that need key files, generate an `agent<ID>.key` for each agent:

```
openssl genrsa -out agent<ID>.key 4096
```

Where `<ID>` is the agent identifier.

1. Generate a CSR for each agent:

```
openssl req -new -key agent<ID>.key -out agent<ID>.csr -subj /C=IN/ST=MH/L=Pune/O=PEM/
↪CN=agent<ID>
```

Where `-subj` is provided as per your requirements. Replace `<ID>` in `CN` with an appropriate agent identifier.

Note

If you prefer to use a single certificate for all PEM agents rather than one per agent, create a common Postgres user and supply this username in place of *ID* . See *Generate a common agent certificate and key pair* .

1. Ask your CA to sign the CSR and generate the agent certificate for you.
2. Copy the certificate and key files on the respective hosts, where `<ID>` matches the `agent_id` in the `/usr/edb/pem/agent/etc/agent.cfg` file.
3. Change the ownership and permission on the new `agent<ID>.crt` and `agent<ID>.key` file:

```
chown root agent<ID>.crt agent<ID>.key
chmod 600 agent<ID>.crt agent<ID>.key
```

1. Back up the old agent certificate and key file:

```
# Running as root
mkdir root/.pem/certs
mv root/.pem/agent<ID>.* root/.pem/certs
```

1. Replace each agent's certificate and key file with the newly generated files:

```
cp agent<ID>.key agent<ID>.crt root/.pem
```

7. Restart the PEM agent service.

   - On Linux:

```
# Running as root
systemctl restart pemagent
```

   - On Windows: Use the Services applet to restart the PEM agent. The PEM agent service is named Postgres Enterprise Manager Agent. Select the service name in the Services dialog box, and select **Restart the service**.

> **Note**
>
> For agents registered after following the preceding process, you can provide a certificate to the agent at the time of registration as shown in the Providing an existing SSL certificate and key .

!!!note If the new agent certificate requires a passphrase, then specify the path to a script that returns the passphrase using the `agent_ssl_passphrase_script` parameter. For more information on the parameter, see *Modifying agent configuration* .

21.5 Testing certificates

If you experience authentication problems, you can use these tests to validate certificates.

21.5.1 Validate a certificate against the root certificate

To check whether a PEM agent certificate is trusted according to the server's `root.crt` , copy both certificates to the same machine. Then execute the following command:

```
openssl verify -verbose -CAfile root.crt  agent1.crt
```

This command returns `agent1.crt: OK` on success or an explanatory message on failure.

21.5.2 Make a test connection to the PEM backend database

To verify whether the agent user can connect using a certificate, as root on the server where the agent is located, execute:

```
PGHOST=<pem_host>
PGPORT=<pem_db_port>
PGUSER=agent<ID>
PGSSLCERT=/root/.pem/agent<ID>.crt
PGSSLKEY=/root/.pem/agent<ID>.key
PGSSLMODE=require
```

(continues on next page)

(continued from previous page)

```
export PGHOST PGPORT PGUSER PGSSLCERT PGSSLKEY PGSSLMODE

<psql_path> -A -t -d pem -c "SELECT version()"
```

Where:

- `<psql_path>` is the full path to the psql executable, for example `/usr/edb/as15/bin/psql` .
- `<pem_host>` is the hostname or IP address of PEM server.
- `<pem_db_port>` is the PEM backend database server port.
- `<ID>` is the ID of the agent you're testing, as defined in the file `/usr/edb/pem/agent/etc/agent.cfg` .

Note

If you used the instructions in *Generate a common agent certificate and key pair* , you must set *PGUSER* to the common agent username.

If the connection succeeds, it returns the Postgres version of the database server. Success means that your certificate is valid and the Postgres user is correctly configured.

Regenerating the server SSL certificates

If the PEM backend database server certificates are near expiring, plan to regenerate the certificates and key files.

Important

PEM performs these steps by default when the certificates are nearing expiry. These instructions are provided for completeness in case you need to manually regenerate the PEM certificates and keys.

To replace the SSL certificates:

1. Stop all running PEM agents, first on the server host and then on any monitored host. - On Linux:

```
# Running as root
systemctl stop pemagent
```

- On Windows: Use the Services applet to stop the PEM agent. The PEM agent service is named Postgres Enterprise Manager Agent. In the Services dialog box, select the service name, and select **Stop the service**.

1. Back up the existing SSL certificates and keys:

```
cd /var/lib/edb/as<x>/data
mkdir certs
mv server.* root.* ca_* certs/
```

1. Use the `openssl` command to generate the `ca_key.key` file:

```
openssl genrsa -out ca_key.key 4096
```

1. Move the `ca_key.key` file to the data directory of the backend server, and change the permissions:

```
mv ca_key.key /var/lib/edb/as<x>/data
chmod 600 /var/lib/edb/as<x>/data/ca_key.key
```

1. Use `ca_key.key` to generate the `ca_certificate.crt` file:

```
openssl req -x509 -nodes -days 3650 -newkey rsa:4096 -keyout ca_key.key -out ca_
↪certificate.crt
```

1. Change the permissions of the `ca_certificate.crt` file:

```
chmod 600 /var/lib/edb/as<x>/data/ca_certificate.crt
```

1. Reuse the `ca_certificate.crt` file as the `root.crt` file:

```
cp /var/lib/edb/as<x>/data/ca_certificate.crt /var/lib/edb/as<x>/data/root.crt
```

1. Change the owner and permissions on the `root.crt` file:

```
chown enterprisedb /var/lib/edb/as<x>/data/root.crt
chmod 600 /var/lib/edb/as<x>/data/root.crt
```

1. Use the `openssl_rsa_generate_crl()` function to create the certificate revocation list `root.crl` :

```
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. Change the ownership and permissions of the `root.crl` file:

```
chown enterprisedb /var/lib/edb/as<x>/data/root.crl
chmod 600 /var/lib/edb/as<x>/data/root.crl
```

1. Use the `openssl` command to generate the `server.key` file:

```
openssl genrsa -out server.key 4096
```

1. Move `server.key` to the data directory of the backend server, and change the ownership and permissions:

```
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. Use the `openssl req` command to create the CSR:

```
openssl req -new -key server.key -out server.csr -subj /C=IN/ST=MH/L=Pune/O=EDB/CN=PEM
```

Where `-subj` is provided as per your requirements. You define `CN` as the hostname/domain name of the PEM server host.

1. Use the `openssl x509` command to sign the CSR and generate a server certificate. Move `server.crt` to the data directory of the backend database server:

```
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
```

Where `-req` indicates the input is a CSR. The `-CA` and `-CAkey` options specify the root certificate and private key to use for signing the CSR.

1. Change the owner and the permissions on the `server.crt` file:

```
chown enterprisedb /var/lib/edb/as<x>/data/server.crt
chmod 600 /var/lib/edb/as<x>/data/server.crt
```

1. Restart the PEM server:

```
systemctl restart edb-as-<x>
```

Restarting the backend database server restarts the PEM server.

1. Regenerate each PEM agent's SSL certificates. For more information, see *Regenerating the agent SSL certificates* .

Regenerating the agent SSL certificates

Important

PEM performs these steps by default when the certificates are nearing expiry. These instructions are provided for completeness in case you need to manually regenerate the PEM certificates and keys.

You need to regenerate the agent certificates and key files:

- If the PEM server certificates are regenerated.
- If the PEM agent certificates are near expiring.

You must regenerate a certificate and a key for each agent interacting with the PEM server and copy it to the agent.

Each agent has a unique identifier that's stored in the pem.agent table of the `pem` database. You must replace the certificate and key files with the certificate or key files that correspond to the agent's identifier.

Prerequisites:

- PEM server has certificates.
- `ca_certificate.crt` and `ca_key.key` are in the data directory of the PEM backend database server.
- `ca_certificate.crt` is the same as `root.crt` .
- `ca_certificate.crt` and `ca_key.key` are valid SSL certificates and keys.

To generate a PEM agent certificate and key file pair:

1. Use psql to find the number of agents and their corresponding identifiers:

```
# Running as enterprisedb
psql -p 5444 -U enterprisedb -d pem --no-psqlrc -t -A -c "SELECT id FROM pem.agent␣
↪WHERE active=true"
```

2. Stop all the running PEM agents:

```
# Running as root
systemctl stop pemagent
```

On Windows, use the Services applet to stop the PEM agent. The PEM agent service is named Postgres Enterprise Manager Agent. In the Services dialog box, select the service name, and select **Stop the service**.

3. After identifying the agents that need key files, generate an `agent.key` for each agent:

```
openssl genrsa -out agent<ID>.key 4096
```

Where `ID` is the agent identifier.

4. Generate a certificate signing request (CSR) for each agent:

```
openssl req -new -key agent<ID>.key -out agent<ID>.csr -subj /C=IN/ST=MH/L=Pune/O=PEM/
↪CN=agent<ID>
```

Where `CN` is the `agent<ID>` .

5. Use the `openssl x509` command to sign the CSR and generate an agent certificate:

```
openssl x509 -req -days 365 -in agent<ID>.csr -CA ca_certificate.crt -CAkey ca_key.
↪key -CAcreateserial -out agent<ID>.crt
```

Where `-req` indicates the input is a CSR. The `-CA` and `-CAkey` options specify the root certificate and private key to use for signing the CSR.

Before generating the next certificate and key-file pair, move the `agent.key` and `agent.crt` files generated in steps 2 and 4 on their respective PEM agent host.

6. Change the permission on the new `agent<ID>.crt` and `agent<ID>.key` files:

```
chmod 600 agent<ID>.crt agent<ID>.key
```

7. Back up the old agent certificate and key files:

```
mkdir root/.pem/certs
mv root/.pem/agent<ID>.* root/.pem/certs
```

8. Replace each agent's certificate and key file with the newly generated files:

```
cp agent<ID>.key agent<ID>.crt root/.pem
```

9. Start the PEM agent service.

    - On Linux:

```
# Running as root
systemctl start pemagent
```

```
- On Windows: Use the Services applet to start the PEM agent. The PEM agent service␣
↪is named Postgres Enterprise Manager Agent. In the Services dialog box, select the␣
↪service name, and select **Start the service**.
```

CHAPTER 22

Managing configuration settings

The PEM web application reads multiple configuration files at startup. These files are located at `/usr/edb/pem/web` on Linux and at `C:\ProgramFiles\edb\pem\server\share\web` on Windows.

- `config.py` _ This is the main configuration file. Don't modify it. You can use it as a reference for configuration settings that one of the following files might be override.
- `config_distro.py` _ This file is read after `config.py` and is intended for packagers to change any settings that are required for their Postgres Enterprise Manager distribution. This typically includes certain paths and file locations. This file is optional. Packagers can create it in the same directory as `config.py` if it's needed.
- `config_local.py` _ This file is read after `config_distro.py` and is intended for end users to change any default or packaging specific settings that they want to adjust to meet local preferences or standards. This file is optional. Create it in the same directory as `config.py` if it's needed.

CHAPTER 23

Troubleshooting agent issues

You can troubleshoot these issues that can occur with the PEM agent.

23.1 Restoring a deleted PEM agent

If an agent was deleted from the `pem.agent` table then you can't restore it. You need to use the pemworker utility to reregister the agent.

If an agent was deleted from the PEM web client but still has an entry in the `pem.agent` table with value of `active = f` , then you can restore the agent:

1. Check the values of the `id` and `active` fields:

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

2. Update the status for the agent to `true` in the `pem.agent` table:

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

Where `x` is the identifier that was displayed in the output of the query used in step 1.

3. Refresh the PEM web client.

The deleted agent is restored. However, the servers that were bound to that agent might appear to be down. To resolve this issue, modify the PEM agent properties of the server to add the bound agent again. After the successful modification, the servers appear as running properly.

23.2 Updating configuration file path (agent status displaying as unknown)

If you move the agent configuration file (`agent.cfg`) from its default location `/usr/edb/pem/agent/etc` to another location, the PEM dashboard might display the agent status as "unknown". In that case, you need to update the value of the agent configuration file path in the `pemagent` service file.

1. Use the following command to modify the `pemagent` service file as a superuser:

```
# Running as superuser
sudo vi  /usr/lib/systemd/system/pemagent.service
```

2. Update the `agent.cfg` file path. For example, on a Redhat host, update the file path:

```
ExecStart=/usr/edb/pem/agent/bin/pemagent -c /usr/edb/pem/agent/etc/agent.cfg
```

3. Reload `systemd` , to update the modified service script:

```
sudo systemctl daemon-reload
```

4. Restart the PEM agent:

```
sudo systemctl restart pemagent
```

23.3 Using the command line to delete a PEM agent with down or unknown status

Using the PEM web interface to delete PEM agents with Down or Unknown status can be difficult if the number of such agents is large. In this situation, you can use the command line interface to delete Down or Unknown agents.

Use the following query to delete the agents that are Down for more than *N* number of hours:

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

Use the following query to delete the agents with an Unknown status:

```
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));
```

CHAPTER 24

Accessing the web interface

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

After the server installation completes, you can open the PEM interface in your choice of browser by navigating to:

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

`ip_address_of_PEM_host` specifies the IP address of the host of the PEM server.

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

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

After providing your credentials, select **Login** to connect to PEM.

The PEM web interface opens, displaying the Global Overview dashboard.

The Browser pane displays a tree that provides access to information about the database objects that reside on each server. The tree expands to display a hierarchical view of the servers and objects that the PEM server monitors.

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

A server can be managed or unmanaged:

- A *managed* server is bound to a PEM agent. The PEM agent monitors the server to which it's bound and performs tasks or reports statistics for display on the PEM dashboards. A managed server has access to extended PEM functionality such as custom alerting. When registering a server, you can also allow PEM to restart a managed server as required.
- An *unmanaged* server isn't bound to a PEM agent. You can create database objects on an unmanaged server, but extended PEM functionality (such as custom alerting) isn't supported on an unmanaged server.

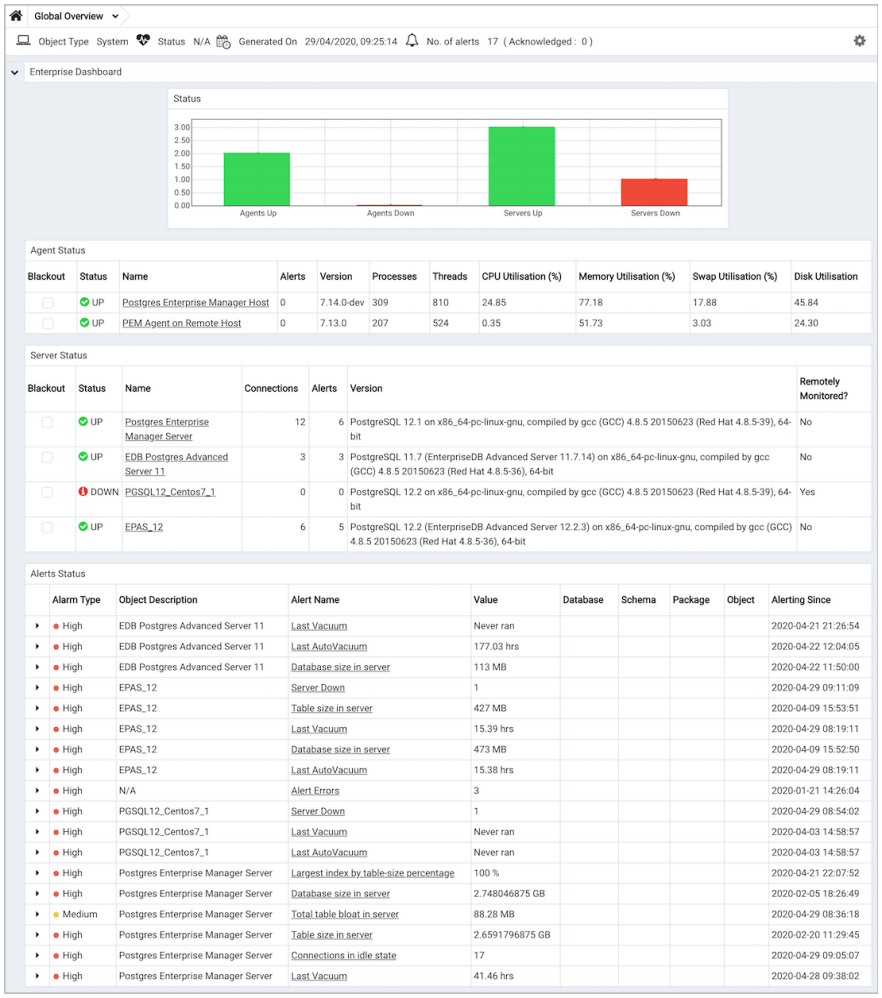


Fig. 1: The Global Overview Dashboard

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

24.1 PEM client object browser

The Browser tree control provides access to information and management options for the database objects that reside on each server. The tree control expands to display a hierarchical view of the servers and objects that are monitored by the PEM server. You can use context menu options, accessed by right-clicking nodes of the tree control, to create new objects and modify and delete existing objects if your role holds the required privileges.

Expand nodes in the tree control to display a hierarchical view of the database objects that reside on a selected server:

- Use the plus sign (+) to the left of a node to expand a segment of the tree control.
- Use the minus sign (-) to the left of a node to close that node.

Context menu options can include one or more of the following selections:

Option	Action
Add named restore point	Create and enter the name of a restore point.
Backup	Open the *Tools menu* dialog box to back up database objects.
Backup Globals	Open the *Tools menu* dialog box to back up cluster objects.
Backup Server	Open the *Tools menu* dialog box to back up a server.
Connect Server	Establish a connection with the selected server.
Create	Access a context menu that provides context-sensitive selections. Your selection opens a Create dialog box for creating a new object.
CREATE Script	Open the *Query information* to edit or view the CREATE script.
Dashboards	Select for quick access to PEM dashboards.
Delete/Drop	Delete the currently selected object from the server.
Disconnect Database	Terminate a database connection.
Disconnect Server	Refresh the currently selected object.
Drop Cascade	Delete the currently selected object and all dependent objects from the server.
Debugging	Access the Debugger tool.
Grant Wizard	Access the *Tools menu* tool.
Maintenance	Open the *Management menu* dialog box to VACUUM, ANALYZE, REINDEX, or CLUSTER.
Management	Access management tasks that are relevant to the node.
Properties	Review or modify the currently selected object's properties.
Refresh	Refresh the currently selected object.
Reload Configuration	Update configuration files without restarting the server.
Restore	Access the *Tools menu* dialog box to restore database files from a backup.
View Data	Use the **View Data** option to access the data stored in a selected table with the **Data Output** tab of the Query tool.

The context-sensitive menus associated with Tables and nested Table nodes provides additional display options.

Option	Action
Import/Export	Open the Import/Export dialog box to import data to or export data from the selected table.
Reset Statistics	Reset statistics for the selected table.
Scripts	Open the *Query information* to edit or view the selected script from the flyout menu.

continues on next page

Table 2 – continued from previous page

Option	Action
Truncate	Remove all rows from a table.
Truncate Cascade	Remove all rows from a table and its child tables.
View First 100 Rows	Access the data grid that displays the first 100 rows of the selected table.
View Last 100 Rows	Access the data grid that displays the last 100 rows of the selected table.
View All Rows	Access the data grid that displays all rows of the selected table.
View Filtered Rows	Access the **Data Filter** popup to apply a filter to a set of data.

24.2 PEM tabbed browser window

The main panel of the PEM client contains a collection of tabs that display information about the object currently selected in the tree control.

The **Dashboard** tab is context sensitive. When you navigate to the **Dashboard** tab from a server group or the PEM Agents node, the EDB Postgres Welcome window opens, where you can:

- Select the **Add New Server** icon to open the *Manually registering a database server* to define a connection to a server.
- Select the **Configure PEM** icon to open the *Editing the PEM server configuration* and modify server parameters.
- Select the **Getting Started** icon to open a new tab, displaying the PEM Getting Started section at the EnterpriseDB website.
- Select the **EDB Website** icon to navigate to the home page of the EnterpriseDB website. The EnterpriseDB website features news about upcoming events and other projects.
- Select the **PostgreSQL Website** icon to navigate to the PostgreSQL project website. The PostgreSQL site features news about recent releases and other project information.
- Select the **EDB Blogs** icon to navigate to the EDB Blog page, where you can review the most-recent employee posts to Postgres related blogs.

Select the name of an agent or server and navigate to the **Dashboard** tab to review session or server activity for the currently selected object.

When opened from the name of an agent or server, the **Dashboard** tab provides a graphical analysis of usage statistics:

- The Server sessions or Database sessions graph displays the interactions with the server or database.
- The Transactions per second graph displays the commits, rollbacks, and total transactions per second that are taking place on the server or database.
- The Tuples In graph displays the number of tuples inserted, updated, and deleted on the server or database.
- The Tuples Out graph displays the number of tuples fetched and returned from the server or database.
- The Block I/O graph displays the number of blocks read from the file system or fetched from the buffer cache (but not the operating system's file system cache) for the server or database.
- The Server activity tabbed panel displays tables that contain session information, session locks, prepared transactions, and configuration.

Navigate to the **Properties** tab to review the properties of the item currently selected in the tree control.

The **SQL** tab displays the SQL code used to generate the object currently selected in the Browser tree control.

The **Statistics** tab displays the statistics gathered for each object on the tree control. The statistics displayed in the table vary by the type of object that's highlighted. Select a column heading to sort the table by the data displayed in the column. Select it again to reverse the sort order. The following table lists some of the statistics that might be displayed.

Panel	Description
PID	The process ID associated with the row.
User	The name of the user that owns the object.
Database	The database name.
Backends	The number of current connections to the database.
Backend start	The start time of the backend process.
Xact Committed	The number of transactions committed to the database in the last week.
Xact Rolled Back	The number of transactions rolled back in the last week.
Blocks Read	The number of blocks read from memory in the last week, in MB.
Blocks Hit	The number of blocks hit in the cache in the last week, in MB.
Tuples Returned	The number of tuples returned in the last week.
Tuples Fetched	The number of tuples fetched in the last week.
Tuples Inserted	The number of tuples inserted into the database in the last week.
Tuples Updated	The number of tuples updated in the database in the last week.
Tuples Deleted	The number of tuples deleted from the database in the last week.
Last statistics reset	The time of the last statistics reset for the database.
Tablespace conflicts	The number of queries canceled because of recovery conflict with dropped tablespaces in database.
Lock conflicts	The number of queries canceled because of recovery conflict with locks in database.
Snapshot conflicts	The number of queries canceled because of recovery conflict with old snapshots in database.
Bufferpin conflicts	The number of queries canceled because of recovery conflict with pinned buffers in database.
Temporary files	The total number of temporary files, including those used by the statistics collector.
Size of temporary files	The size of the temporary files.
Deadlocks	The number of queries canceled because of a recovery conflict with deadlocks in database.
Block read time	The number of milliseconds required to read the blocks read.
Block write time	The number of milliseconds required to write the blocks read.
Size	The size of the selected database, in MB.

The **Dependencies** tab displays the objects on which the currently selected object depends. To ensure the integrity of the database structure, the server makes sure that you don't accidentally drop objects that other objects depend on. You must use `DROP CASCADE` to remove an object on which another object depends.

The **Dependencies** table displays:

- The **Type** field, which specifies the parent object type.
- The **Name** field, which specifies the identifying name of the parent object.
- The **Restriction** field, which describes the dependency relationship between the currently selected object and the parent.

The **Dependents** tab displays a table of objects that depend on the object currently selected in the Browser tree. A dependent object can be dropped without affecting the object currently selected in the Browser tree control.

- The **Type** field specifies the dependent object type.
- The **Name** field specifies the identifying name for the dependent object.
- The **Restriction** field describes the dependency relationship between the currently selected object and the parent.

Navigate to the **Monitoring** tab to access information presented on *Dashboards* . Dashboards display statistical information about the objects monitored by the PEM server.

PEM opens additional tabs when you access PEM functionality through the Management or Tools dialog boxes. Right-click the current tab and select from a context menu that allows you to customize the display for your working style:

- Select **Remove Panel** to remove the currently selected panel.
- Select **Rename Panel** to rename the currently selected panel.
- Select **Detach Panel** to detach the currently selected panel, repositioning it for convenience.
- Select **Add Panel** and select any of the available options to add to the panels.

The PEM client preserves any adjustments when you exit the program. To reset the PEM client to its original format, select **File > Reset Layout**.

24.3 Using chart, graph, and table controls

Use the icons in the upper-right corner of each graphic on a PEM client dashboard to control, download, and customize the charts, graphs, and tables displayed in the PEM client.

Use the **Refresh** icon to display the most recent content available from the PEM probes.

Select the **Download** icon to download a .jpeg or .png image of the chart or graph. By default, the file is in .jpeg format. To save the file as a .png, use the **Personalize** icon to modify the download format.

Select the **Fullscreen** icon to expand the chart or graph to fill the main pane of the PEM client.

Select the **Personalize** icon to modify the display properties of the chart or graph for your session only.

Use the **Information** icon to access information about the chart or graph.

24.3.1 Personalizing a graphic

When you select the **Personalize** icon, the Personalize chart configuration dialog box opens.

Use controls on the Personalize chart configuration dialog box to modify the properties of the graphic:

- Use the **Auto Refresh** control to increase or decrease the number of seconds between refreshes.
- Use the **Auto Refresh** field to specify the number of seconds between updates of the data displayed in the table or chart.
- If applicable, use the **Download as** field to indicate if you want to download a chart as a .jpeg image or a .png image.
- If applicable, use the **Colors** selectors to specify the display colors to use on a chart.
- If applicable, set the **Show Acknowledged Alerts** switch to **Yes** to indicate that you want the table to display alerts that you have acknowledged with a check box in the **Ack'ed** column. Set the field to **No** to indicate for the table to hide any acknowledged alerts. Acknowledged alerts aren't purged from the table content until the time specified in the alert definition passes.

After personalizing the display properties, use the controls in the upper-right hand corner to apply your changes:

- Use the **Delete** icon to reset the properties of the graphic to their default settings. Use the menu to specify whether to apply the change to only this instance of the graphic or to the same graphic when displayed on other dashboards.
- Use the **Save** icon to save your changes to the properties for the graphic. Use the menu to specify to apply the change to only this instance of the graphic or to the same graphic when displayed on other dashboards.

24.4 Browser toolbar

The browser toolbar provides shortcuts for frequently used features, like View Data and the Query tool. This toolbar is visible on the Browser panel. Buttons get enabled/disabled based on the selected browser node.

- Use the **Query tool** button to open the Query tool in the current database context.
- Use the **View Data** button to view/edit the data stored in a selected table.
- Use the **Filtered Rows** button to access the Data Filter popup to apply a filter to a set of data for viewing/editing.

24.5 PEM menu bar

The PEM menu bar provides access to commands and features that you can use to manage your database servers and the objects that reside on those servers. If an option is disabled:

- The database server to which you're currently connected might not support the selected feature.
- The selected menu option might not apply to the current object.
- The role that you used to connect to the server might not have enough privileges to change the selected object.

Context-sensitive menus across the top of the PEM web interface allow you to customize your environment and provide access to the enterprise management features of PEM.

24.5.1 File menu

Use the **File** menu to access the following options.

Option	Action
Preferences	Open the Preferences dialog box to customize your PEM client settings.
Lock Layout	Open a submenu to select the level for locking the UI layout.
Server Configuration	Open the Server Configuration dialog box and update your PEM server configuration settings.
Reset Layout	If a workspace panel is popped out, you can reset it to the default using **Reset Layout** .

24.5.2 Object menu

The **Object** menu is context sensitive. Use the **Object** menu to access the following options.

Option	Action
Create	Access a menu that provides context-sensitive selections.
Refresh	Refresh the currently selected object.
Delete/Drop	Delete the currently selected object from the server.
Connect Server	Open the Connect to Server dialog box to establish a connection with a server.
CREATE Script	Open the Query tool to edit or view the selected script.
Disconnect Server	Refresh the currently selected object.
BART	Access a menu that provides options for removing BART configuration, taking a BART backup, or revalidating the BART configuration.
Clear Saved Password	If you saved the database server password, clear the saved password. Enabled only after password is saved.

continues on next page

Table 5 – continued from previous page

Option	Action
Clear SSH Tunnel Password	If you saved the ssh tunnel password, clear the saved password. Enabled only after password is saved.
Drop Cascade	Delete the currently selected object and all dependent objects from the server.
Hide	Hide the currently selected group. To view hidden groups, enable the **Show hidden groups** option in Preferences.
Properties	Review or modify the currently selected object's properties.
Trigger(s)	Disable or enable triggers for the currently selected table.
Truncate	Remove all rows from a table (**Truncate**) or remove all rows from a table and its child tables (**Truncate Cascade**).
View Data	Access a menu that provides several options for viewing data.
Remove Server	Remove the selected server from the browser tree.
Delete/Drop	Delete the currently selected object from the server.
Connect Database	Connect to selected database.
Count Rows	Count the number of rows of the selected table.
Reset Statistics	Reset the statistics of the selected table.
Scripts	CREATE, DELETE, INSERT, SELECT, and UPDATE script for the selected table.

24.5.3 Management menu

Use the **Management** menu to access the following PEM features.

Option	Action
Audit Manager	Open the Audit Manager and configure auditing on your monitored servers.
Auto Discovery	Open the Auto Discovery dialog box to configure a PEM agent to locate and bind monitored database servers.
Capacity Manager	Open the Capacity Manager dialog box and analyze historical or project future resource usage.
Log Manager	Open the Log Manager dialog box and configure log collection for a server.
Alerts	Open the **Alerts** menu and select **Manage Alerts** to create or modify behavior for alerts.
Manage Charts	Open the **Manage Charts** tab to create or modify PEM charts.
Manage Dashboards	Open the Manage Dashboards dialog box to VACUUM, ANALYZE, REINDEX, or CLUSTER.
Probes	Open the **Probes** menu and select **Manage Probes** to create or modify probes.
Scheduled Tasks	Open the **Scheduled Tasks** tab and review tasks that are pending or recently completed.
Reports	Open the Reports dialog box to generate the alert history report, system configuration report, or core usage report for your server.
Schedule Alert Blackout	Open the Schedule Alert Blackout dialog box and schedule the alerts blackout for your servers and agents.

24.5.4 Dashboards menu

Use the context-sensitive **Dashboards** menu to access dashboards.

Option	Action
Alerts	Open the Alerts dashboard for the selected node.

continues on next page

Table 7 – continued from previous page

Option	Action
Audit Log	Open the Audit Log Analysis dashboard for the selected node.
PGD Admin	Open the PGD Admin dashboard for the selected node.
PGD Group Monitoring	Open the PGD Group Monitoring dashboard for the selected node.
PGD Node Monitoring	Open the PGD Node Monitoring dashboard for the selected node.
Database Server	Open the Database Analysis dashboard for the selected node.
I/O Analysis	Open the I/O Analysis dashboard for the selected node.
Memory	Open the Memory Analysis dashboard for the selected node.
Object Activity	Open the Object Activity Analysis dashboard for the selected node.
Operating System	Open the Operating System Analysis dashboard for the selected node.
Probe Log	Open the Probe Log Analysis dashboard for the selected node.
Server Log	Open the Server Log Analysis dashboard for the selected node.
Session Activity	Open the Session Activity Analysis dashboard for the selected node.
Storage	Open the Storage Analysis dashboard for the selected node.
Streaming Replication	Open the Streaming Replication Analysis dashboard for the selected node.
System Wait	Open the System Wait Analysis dashboard for the selected node.
Session Waits	Open the Session Waits Analysis Dasbhoard for the selected node.

24.5.5 Tools menu

Use the options on the **Tools** menu to access the following features.

Option	Action
Schema Diff	Open the Schema Diff dialog box to compare the schema objects between two database schemas.
Search objects	Open the Search Objects dialog box to search the database objects in a database.
Server	Access the various server-related tools such as Add Named Restore Point, Performance Diagnostics, Queue Server Startup, Queue Server Shutdown, Replace Cluster Primary, Switchover EFM Cluster, and SQL Profiler.
Query tool	Open the Query tool for the currently selected object.
Reload Configuration	Update configuration files without restarting the server.
Pause replay of WAL	Pause replay of the WAL log.
Resume replay of WAL	Resume replay of the WAL log.
Import/Export	Open the Import/Export Data dialog box to import or export data from a table.
Maintenance	Open the Maintenance dialog box to VACUUM, ANALYZE, REINDEX, or CLUSTER.
Backup	Open the Backup dialog box to back up database objects.
Backup Globals	Open the Backup Globals dialog box to back up cluster objects.
Backup Server	Open the Backup Server dialog box to back up a server.
Restore	Open the Restore dialog box to restore database files from a backup.
Grant Wizard	Open the Grant Wizard tool.
Schedule Backup	Open the Schedule Backup dialog box for BART backups.
New ERD Project(Beta)	Open the ERD Tool and start designing your database.
Storage Manager	Open the Storage manager to upload, delete or download the backup files.

24.5.6 Help menu

Use the options on the **Help** menu to access documentation or to review information about the PEM installation.

Option	Action
Quick Search	Type your keywords in the Quick Search field. Typing at least three characters displays all the matching possibilities under Menu items and the relevant documents under Help articles. Select the options under Menu items to perform action of particular functionality or object. Select any of the Help articles to open the help of that topic with highlighted text in a separate window. **Note** : If any of the options under Menu items is disabled, then it provides information by way of an info icon.
REST API Reference	Open the REST API reference.
EDB Website	Open the EDB website in a browser window.
About Postgres Enterprise Manager	Locate versioning and user information for Postgres Enterprise Manager.

24.6 Controlling and customizing charts, graphs, and tables

Use the icons in the upper-right corner of each graphic on a dashboard to control and customize the charts, graphs, and tables displayed in the PEM client for your current user session.

The following table gives the information about icons.

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.
	Expand the chart or graph to full-screen.

CHAPTER 25

Connecting and disconnecting a database server

After defining a server connection, use the `Connect to Server` dialog to authenticate with a server and access the objects stored on the server. To access the dialog, right click on the server name in the PEM client tree control, and select `Connect Server` from the context menu.

If prompted, provide authentication information for the selected server:

- Use the `Password` field to provide the user's password associated with the defined server.
- Select the box next to `Save Password` to instruct the server to save the password for future connections. If you save the password, you are not prompted while reconnecting to the database server with this server definition.

The browser displays a message in a green status bar in the lower right corner when the server connects successfully.

If you receive an error message while attempting a connection, verify that your network is allowing PEM and the host of the database server to communicate. For detailed information about a specific error message, see the Troubleshooting help page.

To review or modify connection details, right-click on the name of the server, and select `Properties...` from the context menu.

25.1 Disconnecting from a server

To disconnect from a server, right-click on the server name in the `Browser` tree control and select Disconnect Server from the context menu. A popup will ask you to confirm that you wish to disconnect the selected server.

CHAPTER 26

Monitoring performance

PEM enables you to implement enterprise-wide performance monitoring of all managed servers.

The top-level dashboard is Global Overview. The Global Overview dashboard presents a status summary of all the servers and agents that are being monitored by the PEM server, a list of the monitored servers, and the state of any currently triggered alerts.

PEM provides a number of benefits not found in any other PostgreSQL management tool:

- Management en masse design _ PEM is designed for enterprise database management and is built to tackle the management of large numbers of servers across geographical boundaries. Global dashboards visually keep you up to date on the up/down/performance status of all your servers.
- Distributed architecture _ The PEM architecture maximizes the ability to gather statistical information and to perform operations remotely on machines regardless of operating system platform.
- Graphical administration _ All aspects of database administration can be carried out in the PEM client using a graphical interface. You can handle server startup and shutdown, configuration management, storage and security control, object creation, performance management, and more from a single console.
- Full SQL IDE _ PEM contains a robust SQL integrated development environment (IDE) that provides ad hoc SQL querying, stored procedure/function development, and a graphical debugger.
- Enterprise performance monitoring _ PEM provides enterprise-class performance monitoring for all managed database servers. Lightweight and efficient agents monitor all aspects of each database server's operations as well as each machine's underlying operating system. They provide detailed statistics to easily navigate performance pages in the interface.
- Proactive alert management _ PEM lets you create performance thresholds for each key metric, such as memory and storage, that are monitored around the clock. Any threshold violation results in an alert sent to a centralized dashboard that communicates the nature of the problem and the actions needed to prevent the situation from jeopardizing the overall performance of the server.
- Simplified capacity planning _ All key performance-related statistics are collected and retained for a specified period in the PEM repository. The Capacity Manager utility allows you to select various statistics and perform trend analysis to understand things such as peak load periods and storage consumption trends. A forecasting mechanism in the tool allows you to also forecast resource usage so you can plan and budget accordingly.

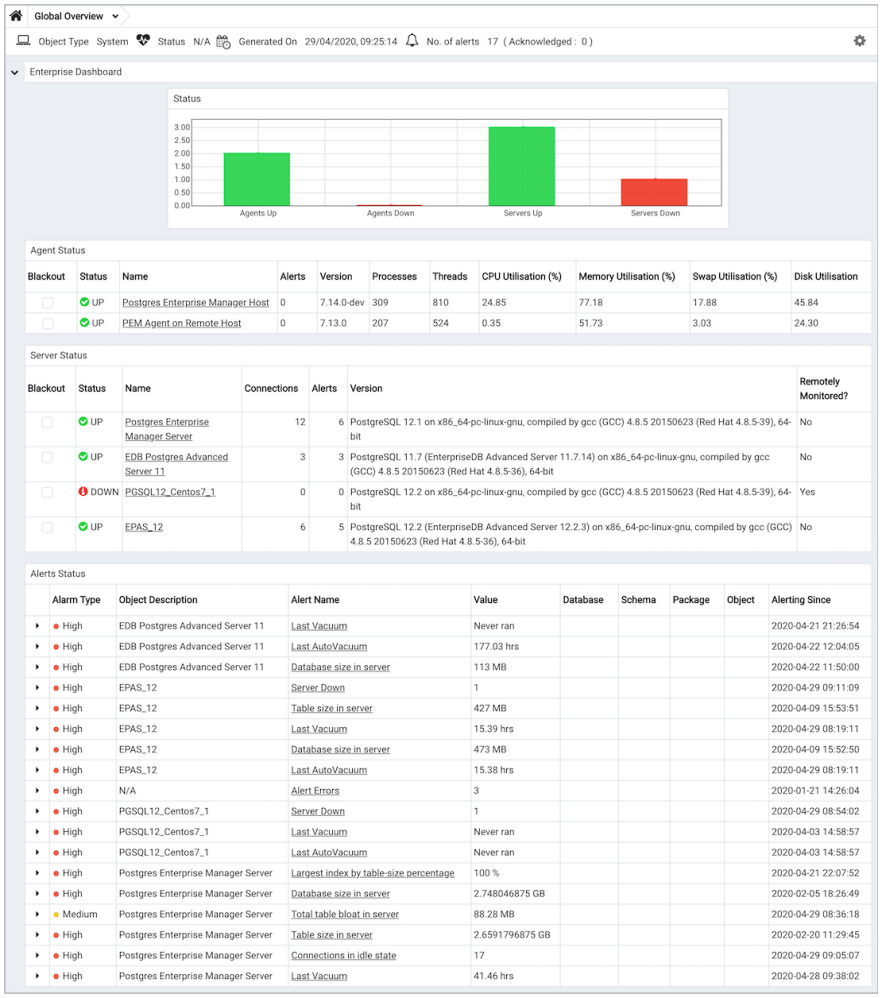


Fig. 1: The Global Overview dashboard

- Audit Manager _ Audit Manager configures audit logging on EDB Postgres Advanced Server instances. You can log activities such as connections to a database, disconnections from a database, and the SQL statements run against a database. You can then use the Audit Log dashboard to filter and view the log.
- Log Manager _ The Log Manager wizard configures server logging parameters with (optional) log collection into a central table. Use the wizard to specify your preference for logging behaviors such as log file rotation, log destination, and the error message severity. Use the Server Log dashboard to filter and review the collected server log entries.
- SQL workload profiling _ PEM contains a SQL profiling utility that allows you to trace the SQL statements that are executed against one or more servers. SQL profiling can be done either in an ad hoc or scheduled manner. You can then filter captured SQL statements so you can easily identify and tune SQL statements that are running poorly.
- Streaming replication monitor _ You can monitor the Streaming Replication dashboard or use options on the PEM client to promote a replica node to the primary node.
- Secure client connectivity _ PEM supports secure client connections through an encrypted SSH tunnel. The full-featured PEM client includes an SSH Tunnel definition dialog box that allows you to provide connection information for a secure connection.
- Wide platform support _ PEM supports most major Linux and Windows platforms.

26.1 Probes

Probe is a scheduled task performed by a Postgres Enterprise Manager (PEM) agent to retrieve performance metrics and information about monitored database objects.

Probes are categorized into the following types:

- *System probes*
- *Creating custom probes*

26.1.1 System probes

PEM features built-in probes at five levels.

- **Agent level** _ Monitors host-specific data such as CPU usage, disk space, memory usage, and network statistics.
- **Server level** _ Tracks global database information including background writer statistics, blocked sessions, session information, and WAL files.
- **Database/Schema level** _ Monitors specific object data such as database size, function statistics, table statistics, and index size.
- **Tool-specific probes** _ Includes probes for Barman (backup), Failover manager (HA), Patroni cluster status, and various replication methods (Streaming, Replication Server).
- **Extension level** _ Monitors specific extension data such as Postgres distributed (PGD) probes .

26.1.2 Customizing probes

You can create, modify, copy, export/import, and delete custom probes to tailor monitoring to your specific needs.

You can modify probe settings using **Management > Probes > Manage Probes** tab.

1. ***Creating custom probes***

Custom probes can be created to monitor specific metrics not covered by built-in probes using three collection methods:

```
- **SQL Query** _ Write custom SQL queries to fetch specific metrics from the␣
↪database. - **WMI** _ Uses WMI to collect data from Windows systems. - **Batch** _␣
↪Uses command-line or shell scripts to gather data from the system. It requires␣
↪`allow_batch_probes` to be enabled in the `agent.cfg` file.
```

2. **Core management tasks**
    - **Modify schedule** _ Customize execution frequency (minutes/seconds) of probes by disabling the **Default** switch.
    - **Enable/Disable probes** _ Toggle the status to **on** or **off** as needed. Disabled probes used in charts displays an icon to re-enable them.
    - **Data retention** _ Specifies how long probe history is stored before automatic deletion.
    - **Copying configurations** _ Use the **Copy probe** button to duplicate probe settings from one object to another except for extension-level probes.
3. **Export/Import and deletion**
    - **Export/Import** _ Use the **Export/Import** buttons to transfer probe configurations between different PEM installations. - **Deletion** _ Custom probes can be deleted using the **Delete** button. System probes cannot be deleted.

Managing and customizing probes

A probe is a scheduled task that returns a set of performance metrics about a specific monitored object. A probe retrieves statistics from a monitored server, database, database object, operating system, or agent.

The **Manage Probes** tab allows administrators to configure the data collection metrics (probes) that Postgres Enterprise Manager (PEM) agents use to monitor databases and servers. This interface provides granular control over what data is collected, how often, and how long it is retained.

Probes overview table

The **Manage Probes** tab displays a list of available probes with the following configurable attributes:

- **Name** _ The descriptive identifier for the probe (e.g., Database Statistics, Tablespace Size).
- **Collection method** _ The protocol used to gather data, such as `SQL` , `WMI` , or `Batch/Shell Script` .
- **Target type** _ The level of the hierarchy being monitored, such as `Server` , `Database` , `Table` , `Schema` , `Index` , `Sequence` , `Function/Procedure` , or so on.
- **Monitoring target** _ The parent entity where the probe is applied such as `Database Server` , `Database` , `Schema` , or `Table` .
- **Interval (Minutes/Seconds)** _ The frequency at which the probe executes (e.g., every 5 minutes or 30 minutes).
- **Enabled?** _ A toggle switch that displays whether a probe is enabled or disabled.
- **Data retention** _ The number of days the collected historical data is stored in the PEM database before being purged.

Key Management Actions

You can use the icons or bars to perform the following tasks:

- **Filter/Search** _ Use the search bar at the top to find specific probes by name.
- **Toggle System Probes** _ Use the **Show System Probes?** switch in the top-right to view or hide built-in default probes.

- **Create Custom Probe** _ Use the plus icon on the top-right toolbar to create a new custom probe.
- **Refresh Probes List** _ Use the refresh icon on the top-right toolbar to reload the list of probes and reflect any recent changes.
- **Save Changes** _ Use the disk icon on the top-right toolbar to save any modifications made to probe configurations.
- **Export/Import** _ Use the outbox arrow and inbox arrow icons in the top-right toolbar to export probe configurations or import the probe configurations from a file.
- **Edit Probe** _ Use the pencil icon on the left of probe name to modify a probe's collection interval or retention period.
- **Delete Probe** _ Use the trash can icon to remove custom probes. You can't delete built-in system probes.

Note

If data from a disabled probe is used in a chart, the chart displays an information icon in the upper-left corner that allows you to enable the probe by clicking the provided link.

Creating custom probes

You can use the **Manage Probes** tab to create a new custom probe or modify an existing custom probe.

From the **Manage Probes** tab, select plus icon on the top-right toolbar to create a custom probe:

1. Use the **General** tab to specify the properties of a new probe:

- Use the **Name** field to provide a name for a new probe.
- Use the **Collection method** field to specify the probe type. From the list, select:
- SQL _ The probe gathers information by way of a SQL statement.
- WMI _ The probe gathers information by way of a Windows Management Instrumentation extension.
- Batch/Shell Script _ The probe uses a command-script or shell-script to gather information.

Before creating a batch probe on a Linux system, you must modify the `agent.cfg` file, setting the `allow_batch_probes` parameter to `true` . Then restart the PEM agent. The `agent.cfg` file is located in `/usr/edb/pem/agent/etc/` directory.

On Windows systems, agent settings are stored in the registry. Before creating a batch probe, modify the registry entry for the `AllowBatchProbes` registry entry and restart the PEM agent. PEM registry entries are located in `HKEY_LOCAL_MACHINE\Software\EnterpriseDB\PEM\agent` .

Batch probes are platform specific. If you specify a collection method of **Batch**, you must specify a platform type in the **Platform** field.

- To invoke a script on a Linux system, you must modify the entry for the `batch_script_user` parameter of the `agent.cfg` file and specify the user to use to run the script. You can specify either a non-root user or root for this parameter. If you don't specify a user, or the specified user doesn't exist, then the script won't run. Restart the agent after modifying the file.
- To invoke a script on a Windows system, set the registry entry for `AllowBatchJobSteps` to `true` and restart the PEM agent.
- Use the **Target type** list to select the object type for the probe to monitor. **Target type** is disabled if **Collection method** is WMI. Configure **Target type** in probes to ensure that the collected metrics are relevant and accurate for the intended level of the system hierarchy. By setting the target type, you can effectively monitor various components, ranging from individual database objects like tables and indexes to server-wide monitoring.

Target type configuration matrix

The following table provides an overview of the target type configurations, their monitoring scope, and the mandatory fields for configuring them.

In this table:

```
-  The **Target type** column determines the object you want to monitor with the␣
↪probe.
```

- The **Execution level** column determines if the monitoring probes execute once per database, per server, per agent, or per extension level.
- The **Mandatory columns** column indicates the columns you must configure in the probe query to ensure the required data is collected.
- The **Probe examples** column provides some existing probes you can explore to better understand how probes are used in practice.

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
Function	Database	schema_name, arg_types, function_type, function_name	function_statistics
Extension	Extension	None	bdr_node_summary

!!!note`1_blank_4_-` The custom probes set to a database or larger target type (including schema, table, index, view, sequence, and functions) collect the information at the database level. *1_blank_4_-* The system probes set to a schema or larger target type collect the information up to the schema level. .. ::
</details>

- Use the **Minutes** and **Seconds** selectors to specify how often the probe collects data.
- Use the **Enabled?** toggle button to specify if the probe is enabled or disabled by default. If you enable the probe, it starts collecting data at the specified interval. If you disable the probe, it doesn't collect data until you enable it.

Note

If data from a disabled probe is used in a chart, the chart displays an information icon in the upper-left corner that allows you to enable the probe by clicking the provided link.

- Use the **Data Retention** field to specify the number of days that gathered information is retained in the probe's history table.
- Use the **Discard from history** toggle button to specify if the server creates a history table for the probe. Select **Yes** to discard probe history or **No** to retain the probe history in a table.
- Use the **Platform** list to specify the type of platform that the probe monitors. This field is enabled only when the **Collection** method is **Batch**.

2. Use the **Columns** tab to define the columns in which to store the probe data. To define a new column:
    1. Navigate to the **Columns** tab and select plus icon in the upper-right corner to add a column.
    2. In the **Name** field, provide a column name.
    3. Select **Edit** (to the left of the new column name) to provide information about the column:

```
-    Provide a descriptive name for the column in the **Name** field.

-    The **Internal name** field isn't enabled for custom probes.

-    Use the **Column type** list to specify if the column is a key column (a␣
↪primary key) or a non-key column. Non-key columns are generally metric items␣
↪(values that can be graphed).

-    Use the **Data type** list to specify the type of data to store in the␣
↪column.

-    Use the **Unit** field to specify the unit of measure that applies to the␣
↪metric stored in the column. This unit is displayed on the y-axis of a custom␣
↪chart or a Capacity Manager chart. This is an optional field.

-    Use the **Graphable?** toggle button to specify if the defined metric can be␣
↪graphed and to make the probe accessible from the Capacity Manager or Manage␣
↪Charts dialog boxes.

-    Use the **Is PIT?** toggle button to specify whether to store the metric by␣
↪point-in-time.
```

Point-in-time metrics change (increase or decrease) at any given point of time. For example, database size is a point-in-time metric. At any given point-in-time, the size of the database is fluctuating. Metrics that aren't point-in-time (also referred to as cumulative metrics) are those whose size always increases over time. For example, Blocks Read and Tuples Read are cumulative metrics. The value stays the same or increases.

```
-    Use the **Calculate PIT?** toggle button to specify for the server to calculate␣
↪a point-in-time value for the metric data. **Calculate PIT** is disabled if **Is␣
↪PIT** is **Yes**.
```

PEM allows you to store point-in time-values of cumulative metrics as well. PEM subtracts the last collected value of a cumulative metric from the current value and stores the difference as a point-in-time value.

3. Use the **Code** tab to specify the default code for the probe to execute:

- If the probe is a SQL probe, you must specify the **SQL SELECT** statement invoked by the probe on the **Code** tab. The column names returned by the query must match the **Internal name** specified on the **Columns** tab. The number of columns returned by the query, as well as the column name, data type, and so on, must match the information specified on the **Columns** tab.

- If the probe is a batch probe, you must specify the shell or .bat script to invoke when the probe runs. The output of the script is as follows:
- The first line must contain the names of the columns provided on the **Columns** tab. Each column name must be separated by a tab (t) character. From the second line onwards, each line must contain the data for each column, separated by a tab character.
- If a specified column is defined as a key column, make sure that the script doesn't produce duplicate data for that column across lines of output. The number of columns specified in the **Columns** tab and their names, data type, and so on must match with the output of the script output.
- If the probe is a WMI probe, you must specify the WMI query as a **SELECT WMI** query. The column name referenced in the **SELECT** statement must be same as the name of the corresponding column specified on the **Column** tab. The column names returned by the query must match the **Internal Name** specified on the **Columns** tab. The number of columns returned by the query, as well as the column name, data type, and so on, must match the information specified on the **Columns** tab.

4. Use the **Alternate Code** tab to provide code to invoke if the probe fires on a specific version of the server. To provide version-specific code, move the **Applies to all database server version?** switch to **No** and select plus icon on the top-right corner to add a specific version. Then, use the **Database Version(s)** list to select a version, and select **Edit** (to the left of the version name) to provide the code to execute when the probe executes on that version.

If you select a database version and leave the **Probe Code** column blank, PEM invokes the code specified on the **Code** tab when the probe executes on a server that matches that version.

When you've finished defining the probe, select **Save** (in the corner of the **Custom Probes** tab) to save the definition and make the probe data available for use on custom charts and graphs.

> **Note**
>
> To prevent agent failures caused by duplicate entries, ensure your custom probe query returns unique rows. If a query returns multiple rows for the same target, you must define at least one Key column to create a unique composite identifier for each row.

Exporting or importing probes

You can export and import probe definitions to share probes across different PEM environments from **Manage Probes** tab. When you export a probe, PEM generates a JSON file with the probe definition. You can use this file to import the probe into another PEM environment. You can also use the generated JSON file as a template to create new probes.

To export the probe:

1. Select any probes from the Probes table.
2. Select outbox arrow icon in the top-right corner of the table.
3. Select **Save File** and then select **OK**, a JSON file with the probe definition is downloaded to your local machine.

To import the probe:

1. Select inbox arrow icon in the top-right corner of the table.
2. Select **Browse** to select the JSON file with the probe code to import.
3. Select **Import**.

After selecting the file to import, you can select the **Skip existing** check box. This option skips the probe if it already exists and displays a message letting you know that the import was skipped.

- If you don't select the check box and the probe already exists, then it doesn't import the probe and an error message is reported.

> **Note**
>
> **Import** can't overwrite the existing probe as it might be configured to retain historical data as per the configured retention policy.

- If you don't select the check box and the probe doesn't exist but the corresponding table in the `pem` schema does, then it imports the probe successfully using the same table.

> **Note**
>
> It's possible for the probe to be deleted and not listed on **Manage Probes** tab and for the table holding the data of that probe to exist in the *pem* schema.

Deleting a custom probe

You can delete only custom probes. To delete a probe:

1. Go to the **Manage Probes** tab.
2. Select the probe name in the probes table, and select trash icon to the left of the probe name.

When you delete a probe, PEM deletes the probe definition and all the historical data collected by that probe. The probe history persists for the length of time specified in the **History Retention** field in the probe definition. During the deletion, the probe definition is deleted and any corresponding tables are dropped from the `pemdata` and `pemhistory` schemas.

System probes are the built-in probes provided by PEM and are part of the PEM schema. You can only modify system probes. If you attempt to delete a system probe, PEM reports an error.

Managing probes using pemworker

From Postgres Enterprise Manager (PEM) version 9.6, you can enable and disable a probe using the pemworker utility using the `--probe-display-name` , `--probe-internal-name` , or `--probe-id` options.

Examples

On Linux

This example shows how to disable a probe using the probe display name:

```
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)
```

This example shows how to enable a probe using the probe display name:

```
export PEM_SERVER_PASSWORD=edb
./pemworker  --pem-user enterprisedb --enable-probe --probe-display-name "database␣
↪size"
__OUTPUT__
Wed Mar 27 02:02:20 2024 INFO: Probe status updated for server (id: 1)
```

On Windows

This example shows how to disable a probe using the probe display name:

```
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)
```

This example shows how to enable a probe using the probe display name:

```
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

The **Probe Configuration** settings are available when modifying a probe within the Postgres Enterprise Manager (PEM) interface.

You can access these settings by navigating to **Management > Probes > Probe configuration**:

- **Name** _ The descriptive name of the specific probe (e.g., Database Statistics, Tablespace Size).
- **Probe enabled?** _ A dropdown menu to toggle the probe state. Selecting `Default` follows the global configuration set for the PEM agent. Choosing `Enable` enables the probe regardless of global settings, while `Disable` disables it.
- **Status** _ Indicates whether the probe is currently enabled or disabled.
  - **Green checkmark** _ The probe is active and enabled.
  - **Red cross mark** _ The probe is inactive and disabled.
- **Execution Frequency** _ The interval at which the probe runs, defined in minutes and seconds.
  - **Use default?** _ A toggle to determine if the probe should use the default execution frequency defined at the PEM agent level. If disabled, you can specify a custom interval.
  - **Time (HH:MM:SS)** _ If the default is disabled, you can manually set the specific time interval for probe execution.
- **History Retention** _ The number of days that the probe's historical data is retained in the PEM database.
  - **Use default?** _ A toggle to determine if the probe should use the default data retention period defined at the PEM agent level. If disabled, you can specify a custom retention period.
  - **In days** _ If the default is disabled, you can manually set the number of days to retain historical data for this probe.

Actions

- **Reset** _ A button to reset the probe's configuration settings back to their default values as defined at the global level. This will override any custom settings you have applied for this probe.
- **Copy probes** _ A button that allows you to copy the configuration of the current probes from the selected object to other objects. This is useful for quickly applying the same settings across multiple probes without having to configure each one individually.

> **Note**
>
> Any modifications made to probe configurations are not instantaneous. Please allow a few seconds for these changes to be fully reflected and synchronized across the system.

Copying probe configurations

You can use the **Copy Probe Configuration** dialog box to copy probe definitions from one monitored object to one or more monitored objects of the same type. To open the **Copy Probe Configuration** dialog box:

1. In the PEM client tree, select the object from which you are copying probes.
2. Select **Management > Probes**.
3. Select **Copy Probe Configurations**.

The **Copy Probe Configurations** dialog box opens that copies the probe definitions from the object from which you opened the **Copy Probe Configuration** dialog box to the locations selected on the tree control.

If you specify a parent node in the **Copy Probe Configuration** tree, PEM copies the probe configurations to each object of the same type that resides under that node in the tree. For example, to copy the probe definitions from one schema to all schemas that reside in a database, select only the parent database of the target schemas. A red warning symbol is displayed to the left of the name of a listed target object if that object is the source of the probe that's being copied.

After you select the target object or objects, select **Configure Probes** to copy the probe definitions to the location selected in the dialog box.

> **Note**
>
> Copy Probe isn't supported for the extension-level probes.

26.2 Alerts

PEM continually monitors registered servers. It compares performance metrics against predefined and user-specified thresholds that specify good or acceptable performance for each statistic. Any deviation from an acceptable threshold value triggers an alert. An alert is a system-defined or user-defined set of conditions that PEM compares to the system statistics. Alerts tell you about conditions on registered servers that require your attention.

26.2.1 Viewing the alerts via Global dashboard

When your system statistics deviate from the boundaries specified for that statistic, the alert triggers. The alert displays a high (red), low (yellow), or medium (orange) severity warning in the left-most column of the Alert Status table on the Global Overview dashboard.

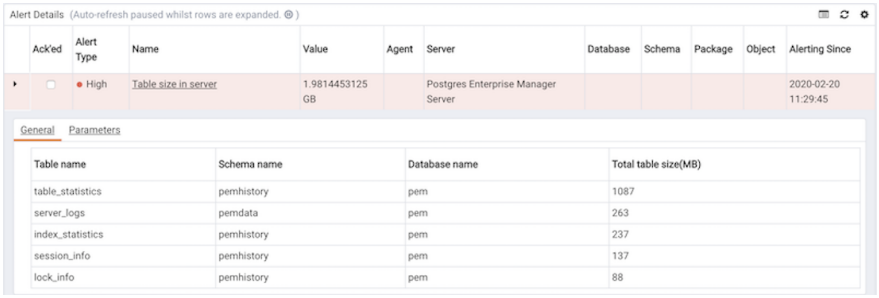
The PEM server includes a number of predefined alerts that are actively monitoring your servers. The alert definition might make details available about the cause of the alert. Select the down arrow to the right of the severity warning to open a dialog box that has details about the condition that triggered the alert.

PEM also provides an interface that lets you create customized alerts. Each alert uses metrics defined on an alert template. An alert template defines how the server evaluates the statistics for a resource or metric. The PEM server includes predefined alert templates, and you can create custom alert templates.

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

Fig. 2: The Alert Status table



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

Fig. 3: Alert details

26.2.2 Viewing the alerts via Alerts dashboard

Use the **Dashboards** menu (on the **Monitoring** tab) to open the Alerts dashboard. The Alerts dashboard shows a summary of the active alerts and the status of each alert.

The Alerts dashboard header shows the date and time that the dashboard was last updated and the number of current alerts.

The Alerts Overview section shows a visual representation of the active alerts and a count of the current high, low, and medium alerts. The vertical bar on the left of the graph provides the count of the alerts displayed in each column. Hover over a bar to display the alert count for the selected alert severity in the upper-right corner of the graph.

The Alert Details table provides a list of the alerts that are currently triggered. The entries appear in order from high severity to low severity. Each entry includes information that lets you identify the alert and recognize the condition that triggered the alert. Select an alert to review detailed information about the alert definition.

The Alert Errors table shows configuration-related errors, such as accidentally disabling a required probe or improperly configuring an alert parameter. You can use the information provided in the Error Message column to identify and resolve the conflict that's causing the error.

26.2.3 Customizing the Alerts dashboard

You can customize tables and charts that appear on the Alerts dashboard. To customize a table or chart, select **Settings** in the upper-right corner.

Use fields on the Personalize Chart Configuration dialog box to provide your display preferences:

- Use the **Auto Refresh** field to specify the number of seconds between updates of the data displayed in the table or chart.

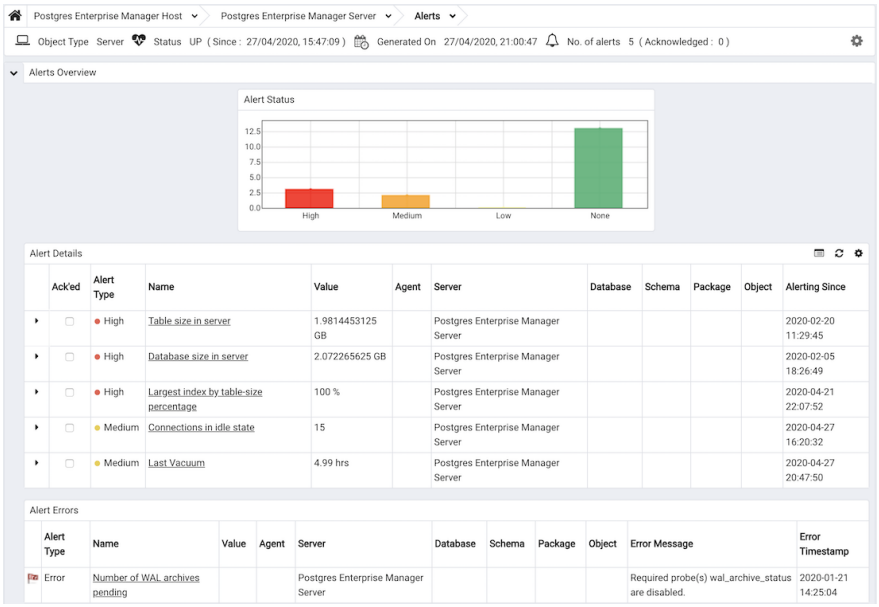


Fig. 4: The Alerts Dashboard

- Use the **Download as** field to indicate whether to download a chart as a JPEG image or as a PNG image.
- Use **Colours selectors** to specify the colors to use on a chart.

To save your customizations, select **Save** (a checkmark) in the upper-right corner. To delete any previous changes and revert to the default values, select **Delete**.

Managing alerts

Use the PEM client's **Manage Alerts** tab to define, copy, or manage alerts. To open the **Manage Alerts** tab, select **Management > Alerts > Manage Alerts**.

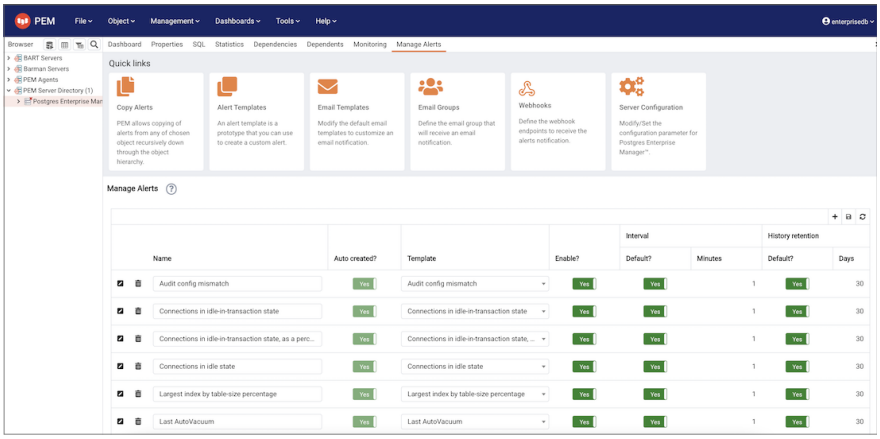


Fig. 5: The Manage Alerts tab

Use the Quick Links toolbar to open dialog boxes and tabs for managing alerts:

- Select **Copy Alerts** to open the Copy Alert Configuration dialog box and copy an alert definition.
- Select **Alert Templates** to open the **Alert Template** tab and modify or create an alert template.

- Select **Email Templates** to open the Email Template dialog box and modify the default email template to customize an email notification.
- Select **Email Groups** to open the **Email Groups** tab and modify or create an email group.
- Select **Webhooks** to open the **Webhooks** tab and create or manage the webhooks endpoints.
- Select **Server Configurations** to open the Server Configuration dialog box and review or modify server configuration settings.

Use the table in the Alerts section of the **Manage Alerts** tab to create new alerts or manage existing alerts.

Alert templates

An alert template is a prototype that defines the properties of an alert. An alert instructs the server to compare the current state of the monitored object to a threshold specified in the alert template to determine if a situation requires administrative attention.

You can use the **Alert Templates** tab to define a custom alert template or view the definitions of existing alert templates. To open the **Alert Templates** tab, select **Management > Alerts > Manage Alerts**. From the **Manage Alerts** tab, on the Quick Links toolbar, select **Alert Templates**.

Use the **Show System Template** list to filter the alert templates that are displayed in the Alert Templates table. From the list, select a level of the PEM hierarchy to view all of the templates for that level.

Defining a new alert template

To define a new alert template, from the **Show System Template** list, select **None**. Then click the plus sign (+) in the upper-right corner of the alert template table. The alert template editor opens.

Use fields on the **General** tab to specify general information about the template:

- Use the **Template name** field to specify a name for the new alert template.
- Use the **Description** field to provide a description of the alert template.
- Use the **Target type** list to select the type of object that is the focus of the alert.
- Use the **Applies to server** list to specify the server type (EDB Postgres Advanced Server or PostgreSQL) to which to apply the alert. You can specify a single server type or **ALL**.
- Use the **History retention** field to specify the number of days to store the result of the alert execution on the PEM server.
- Use the **Threshold unit** field to specify the unit type of the threshold value.
- Use fields in the **Auto create** box to specify for PEM to use the template to generate an automatic alert. If you enable this option, PEM creates an alert when a new server or agent, as specified by the **Target type** list, is added and deletes that alert when the target object is dropped.
- Move the **Auto create?** slider to **Yes** to specify for PEM to create alerts based on the template.
- Use the **Operator** list to select the operator for PEM to use when evaluating the current system values.

Select a greater-than sign (>) to trigger the alert when the system values are greater than the values entered in the **Threshold values** fields.

Select a less-than sign (<) to indicate to trigger the alert when the system values are less than the values entered in the **Threshold values** fields.

- Use the threshold fields to specify the values for PEM to compare to the system values to determine whether to raise an alert. You must specify values for all three thresholds (Low, Medium, and High).

- Use the **Check frequency** field to specify the default number of minutes between alert executions. This value specifies how often the server invokes the SQL code specified in the definition and compares the result to the threshold value specified in the template.

Use the fields on the **Probe Dependency** tab to specify the names of probes referred to in the SQL query specified on the **SQL** tab:

- Use the **Probes** list to select from a list of the available probes.
- To add the probe to the list of probes used by the alert template, select a probe name and select **Add**.
- To remove a probe from the selected probes list, select the probe name and select **Delete**.
- Use the **Parameters** tab to define the parameters to use in the SQL code specified on the **SQL** tab. Select the plus sign (+). Then:
- Use the **Name** field to specify the parameter name.
- Use the **Data type** list to specify the type of parameter.
- Use the **Unit** field to specify the type of unit specified by the parameter.
- Use the **Code** field on the **SQL** tab to provide the text of the SQL query for the server to invoke when executing the alert. The SQL query provides the result against which to compare the threshold value. If the alert result deviates from the specified threshold value, an alert is raised.

In the query, reference parameters defined on the **Parameters** tab sequentially by using the variable param_`x` . The `x` indicates the position of the parameter definition in the parameter list. For example, param_1 refers to the first parameter in the parameter list, param_2 refers to the second parameter in the parameter list, and so on.

The query can also include the following variables:

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}'

- Use the **Detailed Information SQL** field to provide a SQL query to invoke if the alert is triggered. The result set of the query might be displayed as part of the detailed alert information on the Alerts dashboard or Global Overview dashboard.

Note

If the specified query depends on one or more probes from different levels in the PEM hierarchy (server, database, schema, and so on), and a probe becomes disabled, any resulting alerts are displayed as follows:

```
-   If the alert definition and the probe referenced by the query are from the same␣
↪level in the PEM hierarchy, the server displays any alerts that reference the alert␣
↪template on the Alert Error table of the Global Alert dashboard. -   If the alert␣
↪definition and the probe referenced by the query are from different levels of the␣
↪PEM hierarchy, the server displays any triggered alerts that reference the alert␣
↪template on the Alert Details table of the hierarchy on which the alert was defined.
```

To save your changes, select **Done** on the template creation panel. This will return you to the main Alert templates list, where you can then select **Save**. This "bulk save" approach allows you to create or edit multiple templates in one session and commit all your changes with a single click. Once saved, your custom alert templates will be available in the Alerting dialog box for defining new alerts.

Exporting or importing alert templates

To export the alert template:

1. Select any alert template from the **Alert Templates** tab.
2. Select **Export** in the upper-right corner of the table.

This exports the alert template as a JSON file in the downloads folder. You can use the exported JSON file to import the alert template to another PEM server or to keep it as a backup.

To import the Alert Template:

1. On the **Alert Templates** tab, select **Import** in the upper-right corner.
2. To select the JSON file with the code import, select **Browse**, and then select **Import**.
3. After selecting the file to import, you can select the following check boxes:

- **Skip existing** _ Skip the alert template if it already exists.
- **Skip existing dependent probe** _ The alert templates depend on probes. Select this check box to skip the dependent probe if it already exists.

If both the check boxes are selected and the alert template already exists, then it skips importing the alert template.

If you don't select the **Skip existing** check box, select **Skip dependent probe**, and the alert template already exists, then the alert template imports successfully.

If both the check boxes are cleared and the alert template doesn't exist, then it successfully imports the alert template.

Modifying or deleting an alert template

To view the definition of an existing template (including PEM predefined alert templates), use the **Show System Template** list to select the type of object monitored. When you select the object type, the Alert Templates table displays the alert templates that correspond with that object type.

Select a template name in the list, and select **Edit** at the left end of the row to review the template definition.

Use the Alert Templates dialog box to view detailed information about the alert template:

- The **General** tab displays general information.
- The **Probe Dependency** tab lists the names of probes that provide data for the template.
- The **Parameters** tab lists the names of any parameters referred to in the SQL code.
- The **SQL** tab displays the SQL code that defines the behavior of the alert.

To delete an alert template, select the template name in the alert templates table and select **Delete**, located in the upper-right corner of the table. The alert history persists for the time specified in the **History Retention** field in the template definition.

> **Note**
>
> You cannot delete a predefined system alert template. You can only delete custom alert templates that you create.

Predefined alert templates - reference

An alert definition contains a system-defined or user-defined set of conditions that PEM compares to the system statistics. If the statistics deviate from the boundaries specified for that statistic, the alert triggers, and the PEM client displays a warning on the Alerts Overview page and optionally sends a notification to a monitoring user.

The tables that follow list the system-defined alert templates that you can use to create an alert. This list is subject to change and can vary by system.

Templates applicable on agent

Template name	Description	Probe dependency
Load Average (1 minute)	1-minute system load average	load_average
Load Average (5 minutes)	5-minute system load average	load_average
Load Average (15 minutes)	15-minute system load average	load_average
Load Average per CPU Core (1 minutes)	1-minute system load average per CPU core	load_average
Load Average per CPU Core (5 minutes)	5-minute system load average per CPU core	load_average
Load Average per CPU Core (15 minutes)	15-minute system load average per CPU core	load_average
CPU utilization	Average CPU consumption	cpu_usage
Number of CPUs running higher than a	Number of CPUs running at greater than K% utilization threshold	cpu_usage
Free memory percentage	Free memory as a percent of total system memory	memory_usage
Memory used percentage	Percentage of memory used	memory_usage
Swap consumption	Swap space consumed (in megabytes)	memory_usage
Swap consumption percentage	Percentage of swap area consumed	memory_usage
Disk Consumption	Disk space consumed (in megabytes)	disk_space
Disk consumption percentage	Percentage of disk consumed	disk_space
Disk Available	Disk space available (in megabytes)	disk_space
Disk busy percentage	Percentage of disk busy	disk_busy_info
Most used disk percentage	Percentage used of the most utilized disk on the system	disk_space
Total table bloat on host	The total space wasted by tables on a host, in MB	table_bloat, settings
Highest table bloat on host	The most space wasted by a table on a host, in MB	table_bloat, settings
Average table bloat on host	The average space wasted by tables on host, in MB	table_bloat, settings
Table size on host	The size of tables on host, in MB	table_size
Database size on host	The size of databases on host, in MB	database_size
Number of ERRORS in the logfile on agent N in last X hours.	The number of ERRORS in the logfile on agent N in last X hours	N/A

continues on next page

Table 3 – continued from previous page

Template name	Description	Probe dependency
Number of ERRORS in the audit logfile on agent N in last X hours	The number of ERRORS in the audit logfile on agent N in last X hours	N/A
Number of WARNINGS in the logfile on agent N in last X hours	The number of WARNINGS in the logfile on agent N in last X hours	N/A
Number of WARNINGS in the audit logfile on agent N in last X hours	The number of WARNINGS in the audit logfile on agent N in last X hours	N/A
Number of WARNINGS or ERRORS in the logfile on agent N in last X hours	The number of WARNINGS or ERRORS in the logfile on agent N in last X hours	N/A
Number of WARNINGS or ERRORS in audit the logfile on agent N in last X hours	The number of WARNINGS or ERRORS in the logfile on agent N in last X hours	N/A
Package version mismatch	Check for package version mismatch as per catalog	N/A
Total materialized view bloat on host	The total space wasted by materialized views on a host, in MB	mview_bloat, settings
Highest materialized view bloat on host	The most space wasted by a materialized view on a host, in MB	mview_bloat, settings
Average materialized view bloat on host	The average space wasted by materialized views on host, in MB	mview_bloat, settings
Materialized view size on host	The size of materialized views on host, in MB	mview_size
Agent Down	Specified agent is currently down	N/A

Templates applicable on server

Template name	Description	Probe dependency
Audit config mismatch	Check for audit config parameter mismatch	audit_configuration
Average materialized view bloat in server	The average space wasted by materialized views in server, in MB	mview_bloat, settings
Average table bloat in server	The average space wasted by tables in server, in MB	table_bloat, settings
Buffers allocated per second	Number of buffers allocated per second, over the last two probe cycles	background_writer_statistics
Buffers written per second	Number of buffers written per second, over the last two probe cycles	background_writer_statistics
Committed transactions percentage	Percentage of transactions in the server that committed vs. that rolled-back over last N minutes	database_statistics
Conflicting replication slots	Check for slots that are conflicting for a particular server	
Connections in idle state	Number of connections in server that are in idle state	session_info

continues on next page

Table 4 – continued from previous page

Template name	Description	Probe dependency
Connections in idle-in-transaction state	Number of connections in server that are in idle-in-transaction state	session_info
Connections in idle-in-transaction state, as percentage of max_connections	Number of connections in server that are in idle-in-transaction state, as a percentage of maximum connections allowed on server, settings	session_info, settings
Database size in server	The size of databases in server, in MB	database_size
Dead Tuples	Number of estimated dead tuples in server	table_statistics
Dead tuples percentage	Percentage of estimated dead tuples in server	table_statistics
EFM agent down	Failover Manager service is down	efm_cluster_info
EFM cluster failover detected	Failover is triggered when a failover event is recorded for an EFM cluster within the specified time.	failover_manager_node_status
EFM fewer than N nodes active	Triggers when the number of active EFM nodes in a cluster falls below the expected threshold	efm_cluster_node_status
EFM missing primary	EFM cluster is missing the primary node	efm_cluster_node_status
Function Count	Total number of functions in server	oc_function
Highest materialized view bloat in server	The most space wasted by a materialized view in server, in MB	mview_bloat, settings
Highest table bloat in server	The most space wasted by a table in server, in MB	table_bloat, settings
Hot update percentage	Percentage of hot updates in the server over last N minutes	table_statistics
Inactive replication slots	Check for slots that are inactive for a particular server	
Index Scans	Number of index scans in server, over last N minutes	table_statistics
Index size as a percentage of table size	Size of the indexes in server, as a percentage of their tables' size	index_size, oc_index, table_size
InfiniteCache buffers hit percentage	Percentage of block read requests in the server that were satisfied by InfiniteCache, over last N minutes	database_statistics
Largest index by table-size percentage	Largest index in server, calculated as percentage of its table's size, oc_index, table_size	index_size, oc_index, table_size
Largest materialized view (by multiple of unbloated size)	Largest materialized view in server, calculated as a multiple of its own estimated unbloated size; exclude materialized views smaller than N MB	mview_bloat, settings
Largest table (by multiple of unbloated size)	Largest table in server, calculated as a multiple of its own estimated unbloated size; exclude tables smaller than N MB	table_bloat, settings
Last Analyze	Hours since last analyze on the server	table_statistics
Last AutoAnalyze	Hours since last autoanalyze on the server	table_statistics
Last AutoVacuum	Hours since last autovacuum on the server	table_statistics
Last Vacuum	Hours since last vacuum on the server	table_statistics
Live Tuples	Number of estimated live tuples in server	table_statistics
Log config mismatch	Check for log config parameter mismatch	log_configuration
Long-running autovacuums	Number of autovacuum operations in server that have been running for more than N seconds	session_info
Long-running idle connections	Number of connections in the server that have been idle for more than N seconds	session_info

continues on next page

Table 4 – continued from previous page

Template name	Description	Probe dependency
Long-running idle connections and idle transactions	Number of connections in the server that have been idle or transactions idle-in-transaction for more than N seconds	session_info
Long-running idle transactions	Number of connections in the server that have been idle in transaction for more than N seconds	session_info
Long-running queries	Number of queries in server that have been running for more than N seconds	session_info
Long-running transactions	Number of transactions in server that have been running for more than N seconds	session_info
Long-running vacuums	Number of vacuum operations in server that have been running for more than N seconds	session_info
Materialized View Count	Total number of materialized views in server	oc_views
Materialized view size in server	The size of materialized view in server, in MB	mview_size
Multixact ID exhaustion (wraparound)	Check for multixact ID exhaustion (wraparound)	
Number of attacks detected in the last N minutes	The number of SQL injection attacks occurred in the last N minutes	sql_protect
Number of attacks detected in the last N minutes by username	The number of SQL injection attacks occurred in the last N minutes by username	sql_protect
Number of ERRORS in the logfile on server M in the last X hours	The number of ERRORS in the logfile on server M in last X hours	N/A
Number of minutes lag of replica server from primary server	Streaming Replication: number of minutes replica node is lagging behind the primary node	streaming_replication_lag_time
Number of prepared transactions	Number of transactions in prepared state	number_of_prepared_transactions
Number of replica servers lag behind the primary by flush location	Streaming Replication: number of replica servers lag behind the primary by flush location	streaming_replication
Number of replica servers lag behind the primary by replay location	Streaming Replication: number of replica servers lag behind the primary by replay location	streaming_replication
Number of replica servers lag behind the primary by write location	Streaming Replication: number of replica servers lag behind the primary by write location	streaming_replication
Number of users expiring in N days	Number of users whose accounts are expiring in N days	user_info
Number of users whose password expiring in N days	Number of users whose password have expired or are expiring in N days	user_info
Number of WAL archives pending	Streaming Replication: number of WAL files pending to be replayed at replica	wal_archive_status

continues on next page

Table 4 – continued from previous page

Template name	Description	Probe dependency
Number of WAL files	Total number of Write Ahead Log files	number_of_wal_files
Number of WARNINGS in the logfile on server M in the last X hours	The number of WARNINGS in logfile on server M in the last X hours	N/A
Number of WARNINGS or ERRORS in the logfile on server M in the last X hours	The number of WARNINGS or ERRORS in the logfile on server M in the last X hours	N/A
Patroni cluster paused	Patroni: Detects if the Patroni cluster is paused and unavailable for failover.	patroni_cluster_status
Patroni cluster failover detected	Patroni: Failover is triggered when a failover event is recorded for a Patroni cluster within the specified time.	patroni_node_status
Patroni DCS not healthy	Patroni: Detects if the distributed configuration store (etcd) is not healthy.	patroni_cluster_status
Patroni down or out of contact	Patroni: Detects if the Patroni process is not reachable or has failed on a monitored node.	patroni_cluster_status
Patroni no leader detected	Patroni: Detects when Patroni cluster has no leader/master node.	patroni_cluster_status
Patroni timeline mismatch	Patroni: Detects if the node timeline doesn't match the cluster timeline.	patroni_node_status, patroni_cluster_status
Percentage of buffers written by backends	The percentage of buffers written by backends vs. the total buffers written	background_writer_statistics
Percentage of buffers written by backends over the last N minutes	The percentage of buffers written by backends vs. the total buffers backends over last N	background_writer_statistics
Percentage of buffers written by checkpoint	The percentage of buffers written by the checkpoints vs. the total buffers written	background_writer_statistics
PGD Group Raft Consensus	PGD group Raft consensus not working	bdr_monitor_group_raft
PGD Group Raft Leader ID not matching	PGD group Raft leader ID not matching	bdr_group_raft_details
PGD Group versions check	PGD/pglogical version mismatched in PGD group	bdr_monitor_group_raft
PGD worker error detected	PGD worker error detected reported for PGD node	
Replica server lag behind the primary by flush location	Streaming Replication: replica server lag behind the primary by flush location in MB	streaming_replication
Replica server lag behind the primary by replay location	Streaming Replication: replica server lag behind the primary by replay location in MB	streaming_replication
Replica server lag behind the primary by size (MB)	Streaming Replication: replica server lag behind the primary by size in MB	streaming_replication
Replica server lag behind the primary by WAL pages	Streaming Replication: replica server lag behind the primary by WAL pages	streaming_replication

continues on next page

Table 4 – continued from previous page

Template name	Description	Probe dependency
Replica server lag behind the primary by WAL segments	Streaming Replication: replica server lag behind the primary by WAL segments	streaming_replication
Replica server lag behind the primary by write location	Streaming Replication: replica server lag behind the primary by write location in MB	streaming_replication
Sequence Count	Total number of sequences in server	oc_sequence
Sequential Scans	Number of full table scans in server, over last N minutes	table_statistics
Server Down	Specified server is currently inaccessible	N/A
Shared buffers hit percentage	Percentage of block read requests in the server that were satisfied by shared buffers, over last N minutes	database_statistics
Table Count	Total number of tables in server	oc_table
Table size in server	The size of tables in server, in MB	table_size
Tablespace directory missing	Triggers an alert if a tablespace directory is missing or deleted from the operating system.	tablespace_size
Total connections	Total number of connections in the server	session_info
Total connections as percentage of max_connections	Total number of connections in the server as a percentage of maximum connections allowed on server, settings	session_info, settings
Total materialized view bloat in server	The total space wasted by materialized views in server, in MB	mview_bloat, settings
Total table bloat in server	The total space wasted by tables in server, in MB	table_bloat, settings
Transaction ID exhaustion (wraparound)	Check for transaction ID exhaustion (wraparound)	
Tuples deleted	Tuples deleted from server over last N minutes	database_statistics
Tuples fetched	Tuples fetched from server over last N minutes	database_statistics
Tuples hot updated	Tuples hot updated in server, over last N minutes	table_statistics
Tuples inserted	Tuples inserted into server over last N minutes	database_statistics
Tuples returned	Tuples returned from server over last N minutes	database_statistics
Tuples updated	Tuples updated in server over last N minutes	database_statistics
Ungranted locks	Number of ungranted locks in server	blocked_session_info
Unused, non-superuser connections	Number of unused, non-superuser connections on the server, user_info, settings	session_info, user_info, settings
Unused, non-superuser connections as percentage of max_connections	Number of unused, non-superuser connections on the server as a percentage of max_connections of max_connections, user_info, settings	session_info, user_info, settings
View Count	Total number of views in server	oc_views

Templates applicable on database

Template name	Description	Probe dependency
Total table bloat in database	The total space wasted by tables in database, in MB	table_bloat, settings
Largest table (by multiple of unbloated size)	Largest table in database, calculated as a multiple of its own estimated unbloated size; exclude tables smaller than N MB	table_bloat, settings

continues on next page

Table 5 – continued from previous page

Template name	Description	Probe dependency
Highest table bloat in database	The most space wasted by a table in database, in MB	table_bloat, settings
Average table bloat in database	The average space wasted by tables in database, in MB	table_bloat, settings
Table size in database	The size of tables in database, in MB	table_size
Database size	The size of the database, in MB	database_size
Total connections	Total number of connections in the database	session_info
Total connections as percentage of max_connections	Total number of connections in the database as a percentage of maximum connections allowed on server, settings	session_info, settings
Ungranted locks	Number of ungranted locks in database	blocked_session_info
Connections in idle state	Number of connections in database that are in idle state	session_info
Connections in idle-in-transaction state	Number of connections in database that are in idle-in-transaction state	session_info
Connections in idle-in-transaction state,as percentage of max_connections	Number of connections in database that are in idle-in-transaction state, as a percentage of maximum connections allowed on server, settings	session_info, settings
Long-running idle connections	Number of connections in the database that have been idle for more than N seconds	session_info
Long-running idle connections and idle transactions	Number of connections in the database that have been idle or idle-in-transaction for more than N seconds	session_info
Long-running idle transactions	Number of connections in the database that have been idle in transaction for more than N seconds	session_info
Long-running transactions	Number of transactions in database that have been running for more than N seconds	session_info
Long-running queries	Number of queries in database that have been running for more than N seconds	session_info
Long-running vacuums	Number of vacuum operations in database that have been running for more than N seconds	session_info
Long-running autovacuums	Number of autovacuum operations in database that have been running for more than N seconds	session_info
Committed transactions percentage	Percentage of transactions in the database that committed vs. that rolled-back over last N minutes	database_statistics
Shared buffers hit percentage	Percentage of block read requests in the database that were satisfied by shared buffers, over last N minutes	database_statistics
InfiniteCache buffers hit percentage	Percentage of block read requests in the database that were satisfied by InfiniteCache, over last N minutes	database_statistics
Tuples fetched	Tuples fetched from database over last N minutes	database_statistics
Tuples returned	Tuples returned from database over last N minutes	database_statistics
Tuples inserted	Tuples inserted into database over last N minutes	database_statistics
Tuples updated	Tuples updated in database over last N minutes	database_statistics
Tuples deleted	Tuples deleted from database over last N minutes	database_statistics
Tuples hot updated	Tuples hot updated in database, over last N minutes	table_statistics
Sequential Scans	Number of full table scans in database, over last N minutes	table_statistics

continues on next page

Table 5 – continued from previous page

Template name	Description	Probe dependency
Index Scans	Number of index scans in database, over last N minutes	table_statistics
Hot update percentage	Percentage of hot updates in the database over last N minutes	table_statistics
Live Tuples	Number of estimated live tuples in database	table_statistics
Dead Tuples	Number of estimated dead tuples in database	table_statistics
Dead tuples percentage	Percentage of estimated dead tuples in database	table_statistics
Last Vacuum	Hours since last vacuum on the database	table_statistics
Last AutoVacuum	Hours since last autovacuum on the database	table_statistics
Last Analyze	Hours since last analyze on the database	table_statistics
Last AutoAnalyze	Hours since last autoanalyze on the database	table_statistics
Table Count	Total number of tables in database	oc_table
Function Count	Total number of functions in database	oc_function
Sequence Count	Total number of sequences in database	oc_sequence
Index size as a percentage of table size	Size of the indexes in database, as a percentage of their tables' size	table_size
Largest index by table-size percentage	Largest index in database, calculated as percentage of its table's size, oc_index, table_size	index_size, oc_index, table_size
Database Frozen XID	The age (in transactions before the current transaction) of the database's frozen transaction ID	database_frozenxid
Number of attacks detected in the last N minutes	The number of SQL injection attacks occurred in the last N minutes	sql_protect
Number of attacks detected in the last N minutes by username	The number of SQL injection attacks occurred in the last N minutes by last N minutes by username	sql_protect
Queries that have been cancelled due to dropped tablespaces	Streaming Replication: number of queries that have been cancelled due to dropped tablespaces	streaming_replication_db_conflicts
Queries that have been cancelled due to lock timeouts	Streaming Replication: number of queries that have been cancelled due to lock timeouts	streaming_replication_db_conflicts
Queries that have been cancelled due to old snapshots	Streaming Replication: number of queries that have been cancelled due to old snapshots	streaming_replication_db_conflicts
Queries that have been cancelled due to pinned buffers	Streaming Replication: number of queries that have been cancelled due to pinned buffers	streaming_replication_db_conflicts
Queries that have been cancelled due to deadlocks	Streaming Replication: number of queries that have been cancelled due to deadlocks	streaming_replication_db_conflicts
Total events lagging in all slony clusters	Slony Replication: total events lagging in all slony clusters	slony_cluster
Events lagging in one slony cluster	Slony Replication: events lagging in one slony cluster	slony_cluster
Lag time (minutes) in one slony cluster	Slony Replication: lag time (minutes) in one slony cluster	slony_cluster
Total rows lagging in xdb single primary replication	xDB Replication: Total rows lagging in xdb single primary replication	xdb_smr_mmr_replication

continues on next page

Table 5 – continued from previous page

Template name	Description	Probe dependency
Total rows lagging in xdb multi primary replication	xDB Replication: Total rows lagging in xdb multi primary replication	xdb_smr_mmr_replication
Total materialized view bloat in database	The total space wasted by materialized views in database, in MB	mview_bloat, settings
Largest materialized view (by multiple of unbloated size)	Largest materialized view in database, calculated as a multiple of its estimated unbloated size; exclude materialized views smaller than N MB	mview_bloat, settings
Highest materialized view bloat in database	The most space wasted by a materialized view in database, in MB	mview_bloat, settings
Average materialized view bloat in database	The average space wasted by materialized views in database, in MB	mview_bloat, settings
Materialized view size in database	The size of materialized view in database, in MB	mview_size
View Count	Total number of views in database	oc_views
Materialized View Count	Total number of materialized views in database	oc_views

Templates applicable on schema

Template name	Description	Probe dependency
Total table bloat in schema	The total space wasted by tables in schema, in MB	table_bloat, settings
Largest table (by multiple of unbloated size)	Largest table in schema, calculated as a multiple of its own estimated unbloated size; exclude tables smaller than N MB	table_bloat, settings
Highest table bloat in schema	The most space wasted by a table in schema, in MB	table_bloat, settings
Average table bloat in schema	The average space wasted by tables in schema, in MB	table_bloat, settings
Table size in schema	The size of tables in schema, in MB	table_size
Tuples inserted	Tuples inserted in schema over last N minutes	table_statistics
Tuples updated	Tuples updated in schema over last N minutes	table_statistics
Tuples deleted	Tuples deleted from schema over last N minutes	table_statistics
Tuples hot updated	Tuples hot updated in schema, over last N minutes	table_statistics
Sequential Scans	Number of full table scans in schema, over last N minutes	table_statistics
Index Scans	Number of index scans in schema, over last N minutes	table_statistics
Hot update percentage	Percentage of hot updates in the schema over last N minutes	table_statistics
Live Tuples	Number of estimated live tuples in schema	table_statistics
Dead Tuples	Number of estimated dead tuples in schema	table_statistics
Dead tuples percentage	Percentage of estimated dead tuples in schema	table_statistics
Last Vacuum	Hours since last vacuum on the schema	table_statistics
Last AutoVacuum	Hours since last autovacuum on the schema	table_statistics
Last Analyze	Hours since last analyze on the schema	table_statistics
Last AutoAnalyze	Hours since last autoanalyze on the schema	table_statistics
Table Count	Total number of tables in schema	oc_table

continues on next page

Table 6 – continued from previous page

Template name	Description	Probe dependency
Function Count	Total number of functions in schema	oc_function
Sequence Count	Total number of sequences in schema	oc_sequence
Index size as a percentage of table size	Size of the indexes in schema, as a percentage of their table's size	table_size
Largest index by table-size percentage	Largest index in schema, calculated as percentage of its table's size, oc_index, table_size	index_size, oc_index, table_size
Materialized view bloat	Space wasted by the materialized view, in MB	mview_bloat, settings
Total materialized view bloat in schema	The total space wasted by materialized views in schema, in MB	mview_bloat, settings
Materialized view size as a multiple of unbloated size	Size of the materialized view as a multiple of estimated unbloated size	mview_bloat
Largest materialized view (by multiple of unbloated size)	Largest materialized view in schema, calculated as a multiple of its own estimated unbloated size; exclude materialized view smaller than N MB	mview_bloat, settings
Highest materialized view bloat in schema	The most space wasted by a materialized view in schema, in MB	mview_bloat, settings
Average materialized view bloat in schema	The average space wasted by materialized views in schema, in MB	mview_bloat, settings
Materialized view size	The size of materialized view, in MB	mview_size
Materialized view size in schema	The size of materialized views in schema, in MB	mview_size
View Count	Total number of views in schema	oc_views
Materialized View Count	Total number of materialized views in schema	ov_views
Materialized View Frozen XID	The age (in transactions before the current transaction) of the materialized view's frozen transaction ID	mview_frozenxid

Templates applicable on table

Template name	Description	Probe dependency
Table bloat	Space wasted by the table, in MB	table_bloat, settings
Table size	The size of table, in MB	table_size
Table size as a multiple of unbloated size	Size of the table as a multiple of estimated unbloated size	table_bloat
Tuples inserted	Tuples inserted in table over last N minutes	table_statistics
Tuples updated	Tuples updated in table over last N minutes	table_statistics
Tuples deleted	Tuples deleted from table over last N minutes	table_statistics
Tuples hot updated	Tuples hot updated in table, over last N minutes	table_statistics
Sequential Scans	Number of full table scans on table, over last N minutes	table_statistics
Index Scans	Number of index scans on table, over last N minutes	table_statistics
Hot update percentage	Percentage of hot updates in the table over last N minutes	table_statistics
Live Tuples	Number of estimated live tuples in table	table_statistics
Dead Tuples	Number of estimated dead tuples in table	table_statistics
Dead tuples percentage	Percentage of estimated dead tuples in table	table_statistics
Last Vacuum	Hours since last vacuum on the table	table_statistics

continues on next page

Table 7 – continued from previous page

Template name	Description	Probe dependency
Last AutoVacuum	Hours since last autovacuum on the table	table_statistics
Last Analyze	Hours since last analyze on the table	table_statistics
Last AutoAnalyze	Hours since last autoanalyze on the table	table_statistics
Row Count	Estimated number of rows in a table	table_statistics
Index size as a percentage of table size	Size of the indexes on table, as a percentage of table's size	table_size
Table Frozen XID	The age (in transactions before the current transaction) of the table's frozen transaction ID	table_frozenxid

Global templates

Template name	Description	Probe dependency
Agents Down	Number of agents that haven't reported in recently	N/A
Servers Down	Number of servers that are currently inaccessible	N/A
Alert Errors	Number of alerts in an error state	N/A

Defining a new alert

Use the PEM client **Manage Alerts** tab to define, copy, or manage alerts. To open the **Manage Alerts** tab, select **Management > Alerts > Manage Alerts**.

The **Manage Alerts** tab displays a table of alerts that are defined on the object currently selected in the PEM client tree. You can use the Alerts table to modify an existing alert or to create a new alert.

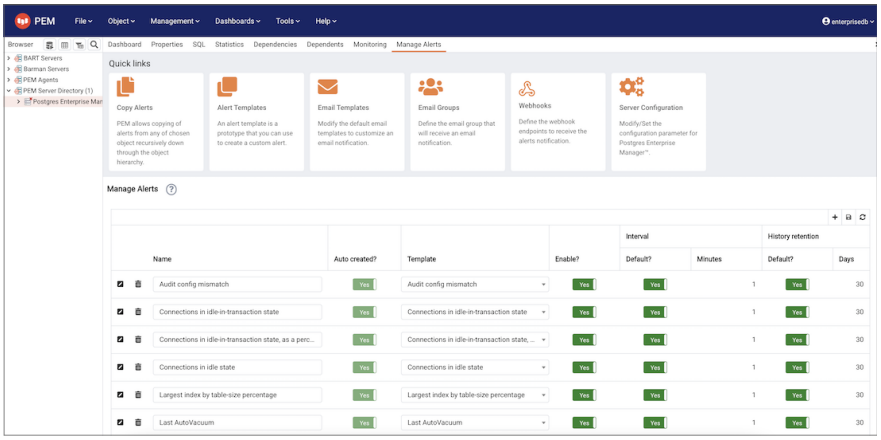


Fig. 6: The Manage Alerts tab

To open the alert editor and create an alert, select the plus sign (+) in the upper-right of the table. The editor opens.

Use the fields on the **General** tab to provide information about the alert:

- Enter the name of the alert in the **Name** field.
- Use the **Template** list to select a template for the alert. An alert template is a function that uses one or more metrics or parameters to generate a value to which PEM compares user-specified alert boundaries. If the value returned by the template function evaluates to a value that's within the boundary of a user-defined alert as specified by the **Operator** and **Threshold values** fields, PEM:

- Raises an alert
- Adds a notice to the Alerts overview display
- Performs any actions specified on the template
- Use the **Enable?** switch to specify if the alert is enabled (**Yes**) or disabled (**No**).
- Use the **Interval** box to specify how often the alert confirms if the alert conditions are satisfied. Use the **Minutes** selector to specify an interval value. Use the **Default** switch to set or reset the **Minutes** value to the default (recommended) value for the selected template.
- Use the **History retention** box to specify the number of days that PEM stores data collected by the alert. Use the **Days** selector to specify the number of days to store the data. Use the **Default** switch to set or reset the **Days** value to the default value (30 days).
- Use controls in the **Threshold values** box to define the triggering criteria for the alert. When the value specified in the **Threshold values** fields evaluates to greater than or less than the system value (as specified with the Operator), PEM raises a Low, Medium or High alert level.
- Use the **Operator** list to select the operator for PEM to use when evaluating the current system values:
- Select a greater-than sign (>) to trigger the alert when the system values are greater than the values entered in the **Threshold values** fields.
- Select a less-than sign (<) to trigger the alert when the system values are less than the values entered in the **Threshold values** fields.
- Use the **Threshold** fields to specify the values for PEM to compare to the system values to determine whether to raise an alert. You must specify values for all three thresholds (Low, Medium, and High).

The Parameter Options table contains a list of parameters that are required by the selected template. The table displays both predefined parameters and parameters for which you must specify a value. You must specify a value for any parameter that displays a prompt in the **Value** column.

PEM can send a notification or execute a script if an alert is triggered or if an alert is cleared. Use the **Notification** tab to specify how PEM behaves if an alert is raised.

Use the **Email notification** box to specify the email group to receive an email notification if the alert is triggered at the specified level. Use the *Email Groups* tab to create an email group that contains the address of the users to notify when an alert is triggered. To access the **Email Groups** tab, select **Email Groups** located in the **Quick Links** menu of the **Manage Alerts** tab.

- To instruct PEM to send an email when a specific alert level is reached, set the slider next to an alert level to **Yes**. Use the list to select the predefined user or group to notify.

You must configure the PEM server to use an SMTP server to deliver email before PEM can send email notifications.

Use the **Webhook notification** box to specify one or multiple endpoints if the alert is triggered at the specified level. Use the *webhooks tab* to create a webhook endpoint to receive the notifications when an alert is triggered. To access the **Webhooks** tab, select **Webhooks** located in the **Quick Links** menu of the **Manage Alerts** tab.

- Set **Enable?** to **Yes** to send the alert notifications to the webhook endpoint.
- Set **Override default configuration?** to **Yes** to set the customized alert levels as per the requirement. Once it's set to **Yes**, all the alert levels are enabled to configure.
- Use the list to select a predefined endpoint to send a notification to for **Low alerts?**, **Medium alerts?**, **High alerts?**, and **Cleared alerts?**.

Use the **Trap notification** options to configure trap notifications for this alert:

- Set **Send trap** to **Yes** to send SNMP trap notifications when the state of this alert changes.
- Set **SNMP Ver** to **v1**, **v2**, or **v3** to identify the SNMP version.

- Use the **Low alert**, **Med alert**, and **High alert** sliders to select the levels of alert to trigger the trap. For example, if you set the slider next to **High alert** to **Yes**, PEM sends a notification when an alert with a high-severity level is triggered.

You must configure the PEM server to send notifications to an SNMP trap/notification receiver before notifications can be sent. For sending SNMP v3 traps, the PEM agent uses 'User Security Model(USM)', which is in charge of authenticating, encrypting, and decrypting SNMP packets.

While sending SNMP v3 traps, the agent creates the `snmp_boot_counter` file. This file is created in the location mentioned by the `batch_script_dir` parameter in `agent.cfg`. If this parameter isn't configured or if the directory isn't accessible due to authentication restrictions, then the file is created in the operating system temporary directory. If that's also not possible, then the file is created in your home directory.

Use the **Nagios notification** box to instruct the PEM server to notify Nagios network-alerting software when the alert is triggered or cleared. For more details, see *Using PEM with Nagios*

- Set the **Submit passive service check result to Nagios** switch to **Yes** to notify Nagios when the alert is triggered or cleared.
- Use the **Script execution** box to optionally define a script that executes if an alert is triggered and to specify details about the script execution.
- Set the **Execute script** slider to **Yes** to instruct PEM to execute the provided script if an alert is triggered.
- Set the **Execute on alert cleared** slider to **Yes** to instruct PEM to execute the provided script when the situation that triggered the alert is resolved.
- Use the **Execute script on** options to indicate for the script to execute on the PEM server or the monitored server.
- In the **Code** field, provide the script for PEM to execute. You can provide a batch/shell script. In the script, you can use placeholders for the following:

`%AlertName%` _ The name of the triggered alert.

`%ObjectName%` _ The name of the server or agent on which the alert was triggered.

`%ThresholdValue%` _ The threshold value reached by the metric when the alert triggered.

`%CurrentValue%` _ The current value of the metric that triggered the alert.

`%CurrentState%` _ The current state of the alert.

`%OldState%` _ The previous state of the alert.

`%AlertRaisedTime%` _ The time that the alert was raised or the most recent time that the alert state was changed.

To invoke a script on a Linux system, you must modify the entry for the `batch_script_user` parameter of the `agent.cfg` file and specify the user to use to run the script. You can either specify a non-root user or root for this parameter. If you don't specify a user or the specified user doesn't exist, then the script doesn't execute. Restart the agent after modifying the file.

To invoke a script on a Windows system, set the registry entry for `AllowBatchJobSteps` to true and restart the PEM agent. PEM registry entries are located in `HKEY_LOCAL_MACHINE\Software\EnterpriseDB\PEM\agent`.

After you define the alert attributes, select **Edit** to close the alert definition editor and then **Save** in the upper-right corner of the Alerts table.

To discard your changes, select **Refresh**. A message prompts you to confirm that you want to discard the changes.

Note

Suppose you need to use the alert configuration placeholder values in an external script. You can do so either by passing them as the command-line arguments or exporting them as environment variables. The external script must have proper execution permissions.

- You can run the script with any of the placeholders as command-line arguments.

For example:

```
#!/bin/bash

bash <path_to_script>/script.sh "%AlertName%  %AlertLevel% %AlertDetails%"
```

- You can define the environment variables for any of the placeholders and then use those environment variables in the script.

For example:

```
#!/bin/bash

export AlertName=%AlertName%
export AlertState=%AlertState%

bash <path_to_script>/script.sh
```

Modifying an alert

Use the Alerts table to manage an existing alert or create a new alert. Select an object in the PEM client tree to view the alerts that monitor that object.

You can modify some properties of an alert in the Alerts table:

- The Alert name column displays the name of the alert. To change the alert name, replace the name in the table and select **Save**.
- The Alert template column displays the name of the alert template that specifies properties used by the alert. You can use the list to change the alert template associated with an alert.
- Use the **Alert enable?** switch to specify if an alert is enabled (**Yes**) or disabled (**No**).
- Use the Interval column to specify how often PEM checks whether the alert conditions are satisfied. Set the **Default** switch to **No** and specify an alternate value, in minutes. Or set the **Default** switch to **Yes** to reset the value to its default setting. By default, PEM checks the status of each alert once every minute.
- Use the **History retention** field to specify the number of days that PEM stores data collected by the alert. Set the **Default** switch to **No** and specify an alternative value in days. Or set the **Default** switch to **Yes** to reset the value to its default setting. By default, PEM stores historical data for 30 days.

After modifying an alert, select **Save** (located in the upper-right corner of the table) to preserve your changes.

To modify other alert attributes, select **Edit** to the left of an alert name to open an editor. The editor provides access to the complete alert definition.

Use the Alert Details dialog box to modify the definition of the selected alert. After you modify the alert definition, select **Save**.

Deleting an alert

To mark an alert for deletion, select the **Delete** button next to the alert name and click **Yes** to confirm. This returns you to the main Alert templates list.

You can repeat this process to mark multiple alerts for removal.

To permanently apply these deletions, select **Save** in the top-right corner. This "bulk save" approach allows you to queue up multiple changes and commit them all with a single click.

Copying an alert

To speed up the deployment of alerts in the PEM system, you can copy alert definitions from one object to one or more target objects.

To copy alerts from an object, select the object in the PEM client tree on the main PEM window. Then, select **Management > Alerts > Copy Alerts**. On the **Manage Alerts** tab, from the Quick Links toolbar, select **Copy Alerts**.

The Copy Alert Configuration dialog box copies all alerts from the object selected in the PEM client tree to the objects selected on the dialog box. Expand the tree to select nodes to specify as the target objects. The tree displays a red warning indicator next to the source object.

To copy alerts to multiple objects at once, select a parent node of the targets. For example, to copy the alerts from one table to all tables in a schema, select the check box next to the schema. PEM copies alerts only to targets that are the same type as the source object.

Select **Ignore duplicates** to prevent PEM from updating any existing alerts on the target objects with the same name as those being copied.

Select **Replace duplicates** to replace existing alerts with alerts of the same name from the source object.

Select **Delete Existing Alerts** to delete all the alerts from the target object and copy all the alerts from the source object to the target object.

Select **Configure Alerts** to copy the alerts from the source object to all objects of the same type in or under those objects selected on the Copy Alert Configuration dialog box.

Schedule an alert blackout

You can use the **Management > Alerts > Schedule Alert Blackout** to schedule an alert blackout for your Postgres servers and PEM agents during maintenance. Alerts aren't raised during a defined blackout period.

To schedule an alert blackout, select **Management > Alerts > Schedule Alert Blackout**.

In the Schedule Alert Blackout dialog box, use the tabs to define the blackout period for servers and agents. On the **Server** tab, to add a row, select the plus sign (+) at the top-right corner.

Use the **Server** tab to provide information about an alert blackout period. After you save the blackout period, you can't edit it.

- Use the **Start time** field to provide the date and time to start the alert blackout.
- Use the **Duration** field to provide the interval for which you want to black out the alerts.
- Use the **Servers** field to provide the server name for which you want to black out the alerts. You can also select multiple servers to black out the alerts for all of those servers.

After providing details, select **Save**. The alerts don't appear on the Alerts dashboard for the scheduled interval of that server.

You can also schedule a blackout period for PEM agents using the **Agent** tab on the dialog box. To add a row, on the **Agent** tab, select the plus sign (+) at the top-right corner.

Use the **Agent** tab to provide the information about an alert blackout period. After you save the blackout period, you can't edit it.

- Use the **Start time** field to provide the date and time to start the alert blackout.
- Use the **Duration** field to provide the interval for which you want to black out the alerts.
- Use the **Agents** field to provide the agent name for which you want to black out the alerts. All server-level alerts for the servers bound to that agent black out.

After providing details, save the details by selecting **Save**. The alerts aren't displayed on the Alert dashboard for the scheduled interval for that PEM agent.

You can select **Clone** from the top-right corner of the dialog box to clone the scheduling of an alert blackout. To create the cloned copy of all the selected servers or agents, select the servers or agents you want to clone, and then select **Clone**. You can edit newly created schedules as needed, and then select **Save**.

Select **Delete** from the top-right corner of the dialog box to remove a scheduled alert blackout. Select the servers or agents and then select **Delete**.

Select a server for which you want to delete the scheduled alert blackout, and then select **Delete**. The server prompts for confirmation before deleting that row.

You can select **Reset** to reset the details on the Alert Blackout dialog box to the default settings. Saved blackouts aren't affected.

You can view the scheduled alert blackout details from the `event_history` table in the `pem` schema once the schedule is executed. For more information, see *Monitoring event history* .

26.3 Managing monitoring settings using profiles

Profiles are a a centralized and reusable way to manage monitoring configurations and alerts across multiple servers and agents.

A profile acts as a single template that bundles a specific set of probe configurations and alert definitions. These are the same configurations currently available through the individual `Manage Probes` and `Manage Alerts` menus for a specific server or agent.

A user must have at least:

- `pem_manage_profile` role to create, update, and delete a profile.
- `pem_database_server_registration` role to assign/unassign profiles to servers or agents.

How profiles work:

- Create a profile with a specific set of probe configurations and alert definitions.
- Assign the profile to one or more servers or agents.
- Once assigned, the servers or agents inherit the monitoring settings defined in the profile immediately.
- Any changes made to the profile automatically propagate to all assigned servers and agents.

Any server or agent that doesn't have a profile assigned to it continues to use its individual probe configurations and alert definitions.

26.3.1 Purpose of using profiles

The primary purpose of using profiles is to significantly reduce the administrative burden associated with managing monitoring configurations for a large number of database instances in PEM.

PEM is designed to manage and monitor many instances from a single platform. When dealing with large numbers of servers, manually ensuring that the configuration (probes and alerts) is correct and consistent across all instances can be a time-consuming and error-prone process.

By centralizing settings into reusable profiles, this feature delivers value by:

- Freeing up time _ Automating configuration management across multiple instances, allowing administrators to focus on optimizing database operations rather than administering the monitoring tool.
- Ensuring consistency _ Guaranteeing that a large fleet of servers adheres to the intended monitoring standards.
- Reducing errors _ Minimizing the chance of human errors that could lead to misconfigurations and undermine PEM's ability to effectively monitor your environment.

26.3.2 Key concepts

There are two types of profiles designed to manage different aspects of monitoring:

- **Agent Profile** _ Manages probe configurations and alert definitions for PEM Agents.
- **Server Profile** _ Manages probe configurations and alert definitions for monitored database servers.

The behavior of each profile type is as follows:

Status	Configuration source	Behavior
No Profile assigned	Default settings	The server or agent uses its individual probe configurations and alert definitions.
Profile assigned	Profile settings	The server or agent inherits the probe and alert settings defined in the assigned profile.

- After assigning a profile to a server or agent, when you modify a profile that is already assigned to agents or servers, the changes are not applied immediately. They are saved as a draft version. Changes only take effect on the assigned servers or agents when you explicitly choose to apply the draft changes.
- You also have the option to discard draft changes if you decide not to apply them, by using `Revert` button.
- The individual `Manage Probes` and `Manage Alerts` pages for a profile-managed server or agent display the current configuration inherited from the assigned profile, but you can't make direct changes on these pages.
- The `Copy Probes` and `Copy Alerts` options are unavailable when attempting to copy configurations from a non-profile server or agent to a profile managed server or agent.

26.3.3 Accessing Manage Profiles

The `Manage Profiles` section provides a central interface for creating, configuring, and deploying a reusable monitoring settings across your PEM environment.

- You can access `Manage Profiles` under the `Configuration` menu in the PEM console. `Manage Profiles` tab works independently of objects selected in the object tree. From here, you can:
- *Creating profiles*
- *Editing profiles*
- *Assigning Profiles*
- *Un-assigning profiles*
- *Deleting profiles*

The main table displays key information about each profile:

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.

Creating profiles

A user with `pem_manage_profile` role or a superuser can create a profile.

On the `Manage Profiles` tab use + to add a new profile:

1. Click the "Add New" button.
2. A dialog box pops-up to select a Target Type _ Server or Agent. Once the target type is selected, it can't be changed for that profile. Click `Next` .
3. A new dialog opens with three tabs of configuration options:

- **General** _ Enter a unique name and optional description for the Profile.
- **Probes** _ Select and configure the probes to include in the Profile. Only server, database, schema, agent, and extension-level probes are available for selection at server level.
- **Alerts** _ Select and configure the alerts to include in the Profile. Only server and agent level alerts are available for selection.

4. After configuring the Profile, click `Save` to create it. The new Profile appears in the `Manage Profiles` table.

Editing profiles

On the `Manage Profiles` tab:

- Click the `Edit icon` next to the profile you want to modify.
- The Profile name can't be updated during editing.
- The Profile description, probe configurations, and alerts can be changed.
- If the Profile has active assignments, the `Apply` and `Revert` buttons become available to deploy or discard draft changes.

Changes will only be synced with all assigned servers and agents when you click the `Apply` button. If you click `Revert` , all changes made to the profile will be reset to their previous status.

Assigning Profiles

A user with `pem_database_server_registration` role can assign profiles to servers or agents.

In the object tree on the PEM console:

- Right-click on a server or an agent.
- Select `Properties` from the context menu.

- In the Properties dialog, navigate to the `General` tab.
- Locate the `Assigned Profile` field.
- Use the dropdown menu to select the desired Profile to assign to the server or agent.
- Click `Save` to apply the changes.

Un-assigning profiles

A user with `pem_database_server_registration` role can unassign profiles from servers or agents.

In the object tree:

- Right-click on a server or an agent.
- Select `Properties` from the context menu.
- In the Properties dialog, navigate to the `General` tab.
- Locate the `Assigned Profile` field.
- Use the dropdown menu to select `<None>` to unassign the Profile from the server or agent.
- Click `Save` to apply the changes.

If you unassign a profile from a server or an agent:

- Probes will reset to their default configurations.
- Alerts will remain, but they will no longer be managed by the Profile.
- You can now manage probes and alerts individually for the server or agent from the respective tabs.

Deleting profiles

On the **Manage Profiles** tab:

1. Select the **Delete icon** next to the profile you want to remove.
2. If the profile has no active assignments, a confirmation dialog appears to finalize the deletion.
3. If the profile is actively assigned to any servers or agents, a **Reassign Profile** dialog appears to prevent the loss of monitoring configurations.
4. In the dialog, use the dropdown menu to select a replacement profile or choose the option to reset the targets to system default configurations.
5. Select **Save** to apply the reassignment and delete the original profile.

The reassignment process applies to both **Server** and **Agent** target types.

Permissions for deletion and reassignment

The ability to delete and reassign profiles depends on your assigned PEM roles:

- Users with the `pem_admin` role can delete profiles, reassign them to existing targets, or reset them to defaults.
- Users with the `pem_manage_profile` role can delete unassigned profiles or delete assigned profiles by resetting them to defaults. They cannot reassign a new profile to existing targets.
- Users holding both `pem_manage_profile` and `pem_database_server_registration` roles have full permission to delete profiles and reassign them to new targets.

26.4 Notifications

PEM can send a notification or execute a script if an alert is triggered or cleared. You can send notifications using the following options:

- *SMTP*
- *Webhooks*
- *SNMP++*
- *Nagios*

26.4.1 SMTP

Important

You must configure the PEM server to use an SMTP server to deliver email before PEM can send email notifications.

Creating an email group

PEM monitors your system for conditions that require attention. You can use an email group to specify the email addresses of users for the server to notify if current values deviate from threshold values specified in an alert definition. An email group can notify multiple users or target specific users during the time periods you specify.

Use the **Email Groups** tab to configure groups of SMTP email recipients. To open the **Email Groups** tab, in the PEM client, select **Management > Alerts > Manage Alerts**. When the **Manage Alerts** tab opens, select **Email Groups** from the Quick Links toolbar.

The **Email Groups** tab displays a list of the currently defined email groups. To modify an existing group, select a group name and select **Edit** at the far left end of the row.

To define a new email group, select the plus sign (+) in the upper-right corner of the **Email Groups** table. Use the Email Groups dialog box to define an email group and its members.

Each row in the email group definition associates a set of email addresses with a specific time period. When an alert is triggered, the server evaluates the times specified in each row and sends the message to those group members whose definitions are associated with the time that the alert triggered.

1. Provide a name for the email group in the **Group Name** field.
2. To open the **Options** tab, select the plus sign (+) in the group members table.
3. Add the member addresses to receive notifications for the time period specified:

- Enter a comma-delimited list of recipient addresses in the **Reply to Addresses** field.
- Enter a comma-delimited list of addresses to receive a copy of the email in the **CC Addresses** field.
- Enter a comma-delimited list of addresses to receive a copy of the email without the knowledge of other recipients in the **Bcc Addresses** field.
- Enter the email address to send the messages from in the **From Address** field.
- Use the **Subject prefix** field to provide a message to add to the start of each subject line when a notification is sent.
- Use the **From Time** and **To Time** selectors to specify the time range for notifications to the group members that are identified on this row. Provide these values in the locale of the PEM client host. The PEM server translates the time into other time zones as required.

1. Select **Add** to add a row to the table, and specify another time period and the email addresses to notify during those hours.
2. When you've finished defining the email group, select **Save**.

After creating the email group, you can use the **Manage Alerts** tab to set up the notification details for an alert to direct notifications to the group.

Deleting an email group

To delete an email group, in the Email Group table, select the name of the group and select **Delete**, located to the left of the group name.

The group name appears in the Email Group table in red. Select **Save** to permanently remove the group from the table.

Customizing email templates

PEM monitors your system for conditions that require user attention and sends email notifications. Use **Email Templates** to customize the email subject and the payload that the server sends if the current values deviate from the threshold values specified by an alert.

These are the default email templates that are customizable:

- Alert detected
- Alert level increased
- Alert level decreased
- Alert cleared
- Alert reminder
- Job success
- Job failure
- Job cancellation
- Job step
- Job step (Database server)

To customize the email template, select the edit button next to the email template name. As needed, edit the **Subject** and **Payload** fields from the available list of placeholders. After modifying the field values, select **Save**.

If any of the default email templates are customized, then a green tick mark is displayed for that template in the `Custom Template?` column on the **Email Templates** tab.

To restore the customized email template as the default template, select the custom template and select the undo button at the top of the **Email Templates** tab.

26.4.2 Webhooks

You must configure the PEM Server to use webhooks to receive notification of alert events on threshold value violations in your configured applications.

Creating a webhook

PEM monitors your system for conditions that require user attention. You can use a webhook to create the endpoints to receive a notification if current values deviate from threshold values specified in an alert definition. Based on the events triggered, PEM sends a notification to multiple webhook endpoints or to specific target webhook endpoints.

Use the **Webhooks** tab to configure endpoint recipients. To open the **Webhooks** tab, select **Management > Alerts > Manage Alerts**. From the **Manage Alerts** tab, on the Quick Links toolbar, select **Webhooks**.

The **Webhooks** tab displays a list of the currently defined recipient applications as endpoints. Select an endpoint and select **Edit** at the far left end of the row to modify an existing endpoint.

To define a new webhook, select the plus sign (+) in the upper-right corner of the table. You can then use the **General** tab to define the basic details of the webhook:

- Provide a name for the webhook in the **Name** field.
- Specify a webhook URL to deliver all the notifications to in the **URL** field.
- Set the request method type used to make the call in the **Request Method** field: **POST** or **PUT**.
- By default, webhooks are enabled. To disable a webhook, set **Enable?** to **No**.

Note

The **Enable?** setting works only if the *enable_webhook* parameter is set to *true* in the *agent.cfg* file. By default, the *enable_webhook* parameter is set to *true* only for the agent running on the PEM server host. For all other agents running on other hosts, you need to set it to *true* manually.

Defining webhook SSL configurations

You can define the webhook SSL parameters in the respective agent configuration file or registry in Windows. You can find the list of webhook SSL parameters in *Modifying agent configuration* . If you add or remove any of the agent configuration parameters, you must restart the agent to apply them.

- On Windows systems, PEM registry entries for webhooks are located in `HKEY_LOCAL_MACHINE\Software\EnterpriseDB\PEM\agent\WEBHOOK` .
- On Linux systems, PEM configuration options for webhooks are stored in the `agent.cfg file` , located by default in `/usr/edb/pem/agent/etc` .

```
[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>
```

Fig. 7: Example - Webhook SSL Parameters in agent.cfg file

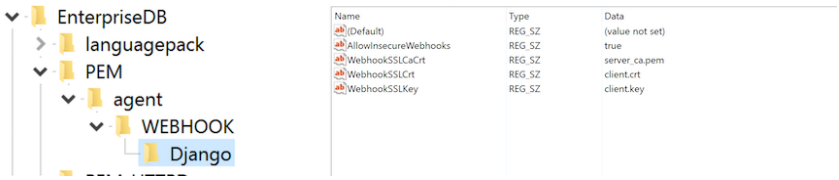


Fig. 8: Example - Webhook SSL Parameters in windows registry

Defining webhook headers

Use the **HTTP Headers** tab to define the header parameters to pass while calling the webhook endpoints:

- Specify all the values as a key and value pair.
- Specify a key parameter in the **Key** field and a value in the **Value** field.

- To add HTTP headers, select the plus sign (+) in the upper-right corner of the HTTP Headers table.
- To delete HTTP headers, select **Delete** to the left of **Key**. The header remains in the list but in strike-through font. Select **Save** to permanently delete the headers.
- To edit the HTTP headers, select **Edit** to the left of **Key**.

Defining webhook payloads

Use the **Payload** tab to define the JSON data to send to the endpoint when an alert is triggered:

- **Type** specifies data to send in the format type, that is, JSON.
- Use **Template** to configure JSON data sent to endpoints. In the template, you can use placeholders for the following:
- `%AlertID%` _ The id of the triggered alert.
- `%AlertName%` _ The name of the triggered alert.
- `%ObjectName%` _ The name of the server or agent on which the alert was triggered.
- `%ObjectType%` _ The type on which the alert was generated.
- `%ThresholdValue%` _ The threshold value reached by the metric when the alert triggered.
- `%CurrentValue%` _ The current value of the metric that triggered the alert.
- `%CurrentState%` _ The current state of the alert.
- `%OldState%` _ The previous state of the alert.
- `%AlertRaisedTime%` _ The time that the alert was raised or the most recent time that the alert state changed.
- `%AgentID%` _ The id of the agent by which the alert was generated.
- `%AgentName%` _ The name of the agent by which the alert was generated.
- `%ServerID%` _ The id of the server on which the alert was generated.
- `%ServerName%` _ The name of the server on which the alert was generated.
- `%ServerIP%` _ The ip or address of the server on which the alert was generated.
- `%ServerPort%` _ The the port of the server on which the alert was generated.
- `%DatabaseName%` _ The name of the database on which the alert was generated.
- `%SchemaName%` _ The name of the schema on which the alert was generated.
- `%PackageName%` _ The name of the package on which the alert was generated.
- `%DatabaseObjectName%` _ The name of the database object, like table name or function name, on which the alert was generated.
- `%Parameters%` _ The list of custom parameters used to generate the alert.
- `%AlertInfo%` _ The detailed database object-level information of the alert.
- Select **Test Connection** to test notification delivery to the mentioned endpoint.

Defining webhook alert levels

Use the **Notifications** tab to specify an alert level for webhook endpoints:

- Set **All alerts** to **Yes** to enable all alert levels to send notifications.

- To send a notification when a specific alert level is reached, set the slider next to an alert level to **Yes**. You must set **All alerts** to **No** to configure an individual alert level.

Example: sending notifications to slack

In Slack, follow the instructions in Getting started with incoming webhooks to:

- Create a Slack app.
- Activate incoming webhooks for that app.
- Add a webhook that posts to a channel or user of your choice.

The new webhook has a unique URL similar to `https://hooks.slack.com/services/x/y/z` . You can now configure PEM to send notifications to this URL.

In PEM, *Creating a webhook* , give it a descriptive name, and copy the URL you obtained earlier to the **URL** field. Ensure that **Request method** is set to **POST** and **Enable?** is set to **Yes**. Set all the sliders under **Alert Notifications** to **Yes**.

Add a header under HTTP headers with the key `Content-Type` and the value `application/json` .

Under **Payload**, delete the default template and specify a template with `text` as the top-level key as in the following example:

```
{"text": "%AlertName% on %ObjectType% %ObjectName% is now %CurrentState%"}
```

You can now test the connection. If it succeeds, PEM issues a notification, and the template you specified appears in your Slack channel as a message.

Save the webhook and continue using PEM as usual. PEM now sends all the alerts to your Slack channel.

Deleting a webhook

To mark a webhook for deletion, in the Webhooks table, select the webhook name and select **Delete** to the left of the name. The alert remains in the list but in strikethrough font.

Delete is a toggle. You can undo the deletion by selecting **Delete** a second time. Select **Save** to permanently delete the webhook definition.

Authenticating webhook requests using OAuth

When OAuth is configured for a webhook endpoint:

1. Before sending a notification, PEM requests an access token from your OAuth provider
2. The token is automatically cached and reused until it expires
3. The access token is included as a `Bearer` token in the request
4. If authentication fails, PEM automatically refreshes the token and retries

Prerequisites

Before configuring OAuth for a webhook, you need:

1. An existing webhook endpoint in PEM
2. OAuth client credentials from your identity provider:

- **Client ID**
- **Client Secret**

- **Token URL** (authorization server endpoint)

3. Optional: **Scopes** and **Audience** values (depending on your provider)
4. Access to a PEM host to run the `pemEncryptor` utility
5. Access to the PEM backend database as a user with `pem_admin` role, or a superuser.

Encrypting credentials

The `client_id` and `client_secret` must be encrypted before storing in the database. Use the `pemEncryptor` utility located at `/usr/edb/pem/bin/` on the PEM server.

Step 1: Encrypt the Client ID

```
export PEM_ENC_PAYLOAD=your-client-id
/usr/edb/pem/bin/pemEncryptor
```

Store the encrypted output for use in the next section.

Step 2: Encrypt the Client Secret

```
export PEM_ENC_PAYLOAD=your-client-secret
/usr/edb/pem/bin/pemEncryptor
```

Store the encrypted output for use in the next section.

Adding OAuth configuration

Use SQL to configure OAuth for an existing webhook. Replace `<encrypted_client_id>` and `<encrypted_client_secret>` with the values obtained from `pemEncryptor` :

```
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
);
```

Configuration parameters

Parameter	Required	Default	Description
webhook_id	Yes	—	The ID of the webhook endpoint
token_url	Yes	—	OAuth token endpoint URL
client_id	Yes	—	OAuth client ID (must be encrypted)
client_secret	Yes	—	OAuth client secret (must be encrypted)
scope	No	NULL	Space-separated OAuth scopes
audience	No	NULL	Target API identifier (auto-derived if not set)
auth_method	No	*client_secret_basic*	How credentials are sent to the token endpoint (see below)

Authentication methods (RFC 6749)

PEM supports two standard client authentication methods defined in RFC 6749 Section 2.3:

Value	Description	When to use
client_secret_basic	Credentials sent via HTTP Basic Auth header (RFC-recommended)	Default — works with most providers
client_secret_post	Credentials sent in the POST request body	Use only when your provider requires it (e.g., some custom OAuth servers)

Note

Most providers (Auth0, Okta, Azure AD, Keycloak) support both methods. AWS Cognito requires *client_secret_basic* . If unsure, use the default.

Updating configuration

```
- - Update scopes
UPDATE pem.webhook_oauth_config
SET scope = api:read api:write,
    modified_on = CURRENT_TIMESTAMP
WHERE webhook_id = 1;

- - Update token URL
UPDATE pem.webhook_oauth_config
SET token_url = https://new-auth.example.com/oauth/token,
    modified_on = CURRENT_TIMESTAMP
WHERE webhook_id = 1;
```

Removing OAuth configuration

To disable OAuth authentication for a webhook:

```
DELETE FROM pem.webhook_oauth_config WHERE webhook_id = 1;
```

26.4.3 SNMP

You must configure the PEM server to send the notifications to an SNMP trap/notification receiver before notifications can be sent. Set the SNMP ver to v1, v2, or v3 to identify the SNMP version.

Example - Configure SNMP V3 traps with net-snmp trap receiver

1. Set `snmp_security_engine_id` to `PEM_SNMP_AGENT` in plain text format in the Server Configuration dialog box.
2. Convert the plain text value to hexadecimal format to use it in `snmptrapd.conf` file. (You can have hexadecimal values of `snmp_security_engine_id` up to 32 octets length).

```
echo PEM_SNMP_AGENT | hexdump -v -e /1 "%02X"
50454D5F534E4D505F4147454E540A
```

1. Set the following parameters in the Server Configuration dialog box:

- **snmp_security_nam** to **pem_snmp_user**
- **snmp_authentication_protocol** to **MD5**
- **snmp_authentication_password** to **pem_auth_pass**
- **snmp_privacy_protocol** to **DES**
- **snmp_privacy_password** to **pem_priv_pass**

1. The `snmptrapd.conf` file has the following values:

```
createUser -e 0x50454D5F534E4D505F4147454E540A pem_snmp_user MD5 pem_auth_pass DES␣
↪pem_priv_pass
authUser log pem_snmp_user
```

Note

The agent must run as root to allow sending notifications with SNMP V3.

26.4.4 Nagios

The PEM server can send a passive alert result to Nagios network-alerting software when a user-defined alert is triggered. To instruct the PEM server to notify Nagios of a triggered alert, you must:

- Enable Nagios notification for each alert that triggers a notification from the PEM server to Nagios. You must configure PEM alerting before you create the `host.cfg` and `services.cfg` files.
- Configure Nagios-related behaviors of the PEM server.
- Create the `host.cfg` and `services.cfg` configuration files.
- If necessary, modify the Nagios configuration file and restart the Nagios server.
- Install the PEM agent on the system where the Nagios server is installed and register it with the PEM Server. Set `enable_nagios` configuration to `true` in the `agent.cfg` file for that agent, and restart the agent service.

After configuring the server to enable Nagios alerting, any triggered alerts send a passive check result to the Nagios service. The syntax of a passive alert is:

```
<timestamp> PROCESS_SERVICE_CHECK_RESULT; <host_name> ; <service_name> ;
<service_status> ;
```

Where:

`timestamp` is the date and time that the alert was triggered.

`host_name` is the name of the server or agent.

`service_name` is the name of the alert.

`service_status` is the numeric service status value:

- 0 if the service status is OK
- 1 if the service status is WARNING
- 2 if the service status is CRITICAL
- 3 if the service status is UNKNOWN

The PEM server uses the following rules to evaluate the service status:

- If the PEM alert level is CLEARED, the warning message reads OK.
- If the PEM alert level is LOW, the warning message reads WARNING.
- If the `is_nagios_medium_alert_as_critical` flag (specified in the PEM server configuration dialog box) is set to FALSE and the alert level MEDIUM, the warning message reads WARNING.
- If the `is_nagios_medium_alert_as_critical` flag (specified in the PEM server configuration dialog box) is set to TRUE and the alert level is MEDIUM, the warning message reads CRITICAL.
- If the PEM alert level is HIGH, the warning message reads CRITICAL.

Enabling Nagios notification for an alert

The PEM server maintains a unique set of notification properties for each enabled alert. Use the **Notification** tab of the **Manage Alerts** tab to specify that, when triggered, a given alert sends an alert notice to Nagios.

To modify the notification properties of an alert, right-click the name of the object monitored by the alert, and select **Management > Alerts > Manage Alerts**. On the **Manage Alerts** tab, select **Edit** to the left of the alert name in the Alerts list. When the edit page opens, select the **Notification** tab.

To enable Nagios notification, move the slider next to **Submit passive service check result to Nagios** to **Yes**. Then select **Save**.

Configuring Nagios-related behavior of the PEM Server

You can use the Server Configuration dialog box to provide information about your Nagios configuration to the PEM server. To open dialog box, select **Management > Server Configuration**.

Four server configuration parameters specify information about your Nagios installation and PEM server behavior related to Nagios:

- Use the `nagios_cmd_file_name` parameter to specify the location of the Nagios pipeline file that receives passive check alerts from PEM. The default value of this parameter is `/usr/local/nagios/var/rw/nagios.cmd` . If your `nagios.cmd` file resides elsewhere, specify the file location in the **Value** field.
- Move the slider in the `nagios_enabled` parameter to **Yes** to instruct the PEM server to send passive check alerts to Nagios.
- Use the `nagios_medium_alert_as_critical` slider to specify the warning severity that the PEM server passes to Nagios if a medium alert is triggered:
- If the `is_nagios_medium_alert_as_critical` flag is set to **FALSE** and the alert level is **MEDIUM**, the warning message reads WARNING.

- If the `is_nagios_medium_alert_as_critical` flag is set to **TRUE** and the alert level is **MEDIUM**, the warning message reads CRITICAL.
- Use the `nagios_spool_retention_time` parameter to specify the number of days of notification history to store on the PEM server. The default value is 7 days.

After modifying parameter values, select **Save** in the upper-right corner of the dialog box.

Creating the hosts.cfg and services.cfg file

The `templates.cfg` file (by default, located in `/usr/local/nagios/etc/objects`) specifies the properties of a generic host and generic service. The properties specify the parameters used in the `hosts.cfg` and `services.cfg` files.

In most cases (when PEM is installed in a default configuration), you don't need to modify the `templates.cfg` file before creating the `hosts.cfg` and `services.cfg` files. If necessary, you can modify the `templates.cfg` file to specify alternative values for parameters or to create new templates.

Before modifying the Nagios configuration file, use the following command to create a `hosts.cfg` file that contains information about the PEM hosts that reside on the local system:

```
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
```

Then, use the following command to create a `services.cfg` file that contains information about the PEM services that reside on the local system:

```
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
```

If you want to use a `custom template.cfg` file entry, specify the entry name in place of `generic-host` or `generic-service` in these commands.

Modifying the Nagios configuration file

After creating the `host.cfg` and `services.cfg` files, you must specify their location in the Nagios configuration file (by default, `/usr/local/nagios/etc/nagios.cfg`). Modify the configuration file, adding entries that specify the location of the files:

```
cfg_file=/usr/local/etc/objects/hosts.cfg
```

```
cfg_file=/usr/local/etc/objects/services.cfg
```

You can use the following command to confirm that Nagios is properly configured:

```
/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg
```

After confirming that Nagios is configured correctly, restart the Nagios service:

```
/usr/local/nagios/bin/nagios -d /usr/local/nagios/etc/nagios.cfg
```

26.5 Capacity Manager

PEM's Capacity Manager analyzes collected statistics (metrics) to generate a graph or table that displays the historical usage statistics of an object. It can also project the anticipated usage statistics for an object. You can configure Capacity Manager to collect and analyze metrics for a specific host, server, database, or database object.

You can tailor the content of the Capacity Manager report by choosing:

- Specific metrics to include in the report
- The time range over which the metrics were gathered

- A high or low threshold for the metrics analyzed

You can also specify a start and end date for the Capacity Manager report. If the end date of the report specifies a time in the future, Capacity Manager analyzes the historical usage of the selected object to extrapolate the projected object usage in the future.

To open Capacity Manager, in the PEM client, select **Management > Capacity Manager**. The Capacity Manager wizard opens, displaying a tree on the **Metrics** tab. On the **Metrics** tab, expand the tree to review the metrics for the node that you want to analyze. Select the metric to include it in your report.

Capacity Manager uses the aggregation method specified with the **Aggregation** list. Capacity Manager uses the aggregation method to evaluate and plot the metric values. Select from:

- **Average** _ Use the average of the values recorded during the time period.
- **Maximum** _ Use the maximum value recorded during the time period.
- **Minimum** _ Use the minimum value recorded during the time period.
- **First** _ Use the first value recorded during the time period.

To remove a metric from the Capacity Manager report, clear the box to the left of the name of a metric.

Move the slider next to **Graph/chart metrics individually?** to **Yes** to produce a separate report for each metric selected on the **Metrics** tab. If the option is set to **No**, all selected metrics are merged into a single graph or table.

Select **Generate** to display the report onscreen (accepting the default configuration options), or use the **Options** tab to customize sampling boundaries, report type, and report destination. The times displayed on the **Options** tab are from the time zone where the PEM client resides.

Use the fields in the **Time Period** box to define the boundaries of the Capacity Manager report:

- Use the **Period** list to select the type of time period to use for the report. You can select:

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.

After specifying the type of time period for the report, select from other options in the **Time Period** box to define the time period for the report:

- Use the date and time selectors next to the **Start time** field to specify the starting date and time of the sampling period. Or select the number of historical days of data to include in the report. The date and time specified in the **Start time** field can’t be later than the current date/time.

By default, Capacity Manager selects a start time one week prior to the current date and time.

- The end boundary for the report can be a time, a number of days in the future, or the point at which a selected metric reaches a specified threshold value. Use the date and time selectors next to **End time** to specify an end boundary for the report. Or select the number of extrapolated days of data to include in the report. The time specified in the **End time** field must be later than the time specified in the **Start time** field.

If you select an end date and time in the future, Capacity Manager uses historical usage information to extrapolate anticipated future usage. Since the projected usage is based on the sampling of historical data, the accuracy of the future usage trend improves with a longer sampling period.

To specify a threshold value, use the list in the **Threshold** field to select a metric, an operator (**Exceeds** or **Falls below**), and a target value for the metric. If you choose to define the end of the report using a threshold, the Capacity Manager report terminates when the value for the selected metric exceeds or falls below the specified value.

The `cm_max_end_date_in_years` configuration parameter defines a default time value for the end boundary of a Capacity Manager report. If you specify a threshold value as the end boundary of a report and the anticipated usage of the boundary isn't met before the maximum time passes, the report terminates at the time specified by the `cm_max_date_in_years` parameter. By default, `cm_max_end_date_in_years` is 5. You can use the Server Configuration dialog box to modify the value of `cm_max_end_date_in_years` .

The fields in the **Report** box specify the report type and destination. Use the **Include on report** options to specify the type of report produced by Capacity Manager:

- Select **Graph** to display the report in the form of a line graph in the PEM client window.
- Select **Table of data** to display a table containing the report data in the PEM client window.
- Select **Graph and table of data** to display both a line graph and a data table in the PEM client window.

Use the **Report destination** options to set where to display or save the report:

- Select **New tab** to display the report on a new tab in the PEM client. You must select **New tab** to display the first generation of a Capacity Manager report. For subsequent reports, you can select **Previous tab**.
- Select **Previous tab** to reuse a previously opened tab when displaying the report.
- Select **Download the report as a file** and specify a file name to write the report to the specified file.
- For **Report Type?**, select **HTML** or **JSON** type (version 9.6 and later).

After you specify the report boundaries and select the type and destination of the Capacity Manager report, select **Generate** to create the report.

You can review a Capacity Manager HTML report with any web browser that supports scalable vector graphics .png). Browsers that don't support .png can't display a Capacity Manager graph and might include unwanted characters.

26.5.1 Capacity Manager templates

After defining a report, you can save the definition as a template for future reports. All PEM users can access Capacity Manager report templates. To save a report definition as a template:

1. Use the **Metrics** and **Options** tabs to define your report.
2. Select **Save**.
3. In the **Save Template** dialog box, Provide a report name in the **Title** field and select a location to store the template in the tree.
4. Select **OK**.

When creating a report, you can use **Load Template** to browse and open an existing template. Once opened, you can modify the report definition and save it again, either as a new template or by overwriting the original template.

Select **Manage Templates** to open a dialog box that lets rename or remove unwanted templates.

26.5.2 Capacity Manager metrics - reference

The Capacity Manager metrics available vary by platform and are subject to change. The available metrics can include the metrics described in the table.

Metric name	Description
# Dead Tuples	The number of dead tuples in the selected table.
# Dead Tuples+	The cumulative number of dead tuples in the selected table.
# Heap Tuples Fetched by Index Scans	The number of heap tuples fetched by index scans.
# Heap Tuples Fetched by Index Scans	The cumulative number of heap tuples fetched by index scans.
# Idle Backends+	The cumulative number of currently idle backend clients.
# Index Scans	The number of index scans performed on the specified object.
# Index Scans+	The cumulative number of index scans performed on the specified object.
# Index Tuples Read	The number of index tuples read.
# Index Tuples Read+	The cumulative number of index tuples read.
# Live Tuples	The number of tuples visible to transactions.
# Live Tuples+	The cumulative number of tuples visible to transactions.
# Pages Estimated by ANALYZE	The number of pages estimated by ANALYZE.
# Pages Estimated by ANALYZE+	The cumulative number of pages estimated by ANALYZE.
# Sequential Scans	The number of sequential scans performed on the specific table.
# Sequential Scans+	The cumulative number of sequential scans performed on the specific table.
# Sequential Scan Tuples	The number of tuples sequentially scanned in the specific table.
# Sequential Scan Tuples+	The cumulative number of tuples sequentially scanned in the specific table.
# Tuples Deleted	The number of tuples deleted.
# Tuples Deleted+	The cumulative number of tuples deleted.
# Tuples Estimated by ANALYZE	The number of live (visible) tuples estimated by ANALYZE.
# Tuples Estimated by ANALYZE+	The cumulative number of live tuples estimated by ANALYZE.
# Tuples HOT Updated	The number of tuples HOT updated. In a HOT update, the new tuple resides in the same block as the original tuple and the tuples share an index entry.
# Tuples HOT Updated+	The cumulative number of tuples HOT updated.
# Tuples Inserted	The number of tuples inserted into the specified table.
# Tuples Inserted+	The cumulative number of tuples inserted into the specified table.
# Tuples Updated	The number of tuples updated in the selected table.
# Tuples Updated+	The cumulative number of tuples updated in the selected table.
Blocks Hit	The number of blocks found in the cache.
Blocks Hit+	The cumulative number of blocks found in the cache.
Blocks Read	The number of blocks read.
Blocks Read+	The cumulative number of blocks read.
Blocks Read from InfiniteCache	The number of blocks read from InfiniteCache.
Blocks Read from InfiniteCache+	The cumulative number of blocks read from InfiniteCache.
Blocks Written	The number of blocks written.
Blocks Written+	The cumulative number of blocks written.
Buffers Allocated	The number of buffers allocated.
Buffers Allocated+	The cumulative number of buffers allocated.
Buffers Written - Backends	The number of buffer blocks written to disk by server processe (processes connected to a client application).
Buffers Written - Backends+	The cumulative number of buffer blocks written to disk by server processes.

continues on next page

Table 14 – continued from previous page

Metric name	Description
Buffers Written - Checkpoint	The number of blocks written to disk by the checkpoint process.
Buffers Written - Checkpoint+	The cumulative number of blocks written to disk by the checkpoint process.
Buffers Written - Cleaning Scan	The number of blocks written to disk by the autovacuum process.
Buffers Written - Cleaning Scan+	The cumulative number of blocks written to disk by the autovacuum process.
Bytes Received (KB)	The number of bytes received from the client (in kilobytes).
Bytes Received (KB)+	The cumulative number of bytes received (in kilobytes).
Bytes Sent (KB)	The number of bytes sent to the client (in kilobytes).
Bytes Sent (KB)+	The cumulative number of bytes sent (in kilobytes).
Checkpoints - Timed	The number of checkpoint operations triggered by the checkpoint interval.
Checkpoints - Timed+	The cumulative number of checkpoint operations triggered by the checkpoint interval.
Checkpoints - Untimed	The number of checkpoint operations triggered by checkpoint size.
Checkpoints - Untimed+	The cumulative number of checkpoint operations triggered by checkpoint size.
Database Size (MB)	The size of the specified database (in megabytes).
Free RAM Memory	The amount of free RAM memory (in megabytes).
Free Swap Memory	The amount of free swap space on disk (in megabytes).
Heap Blocks Hit	The number of heap blocks found in the cache.
Heap Blocks Hit+	The cumulative number of heap blocks found in the cache.
Heap Blocks Read	The number of heap blocks read.
Heap Blocks Read+	The cumulative number of heap blocks read.
Index Blocks Hit	The number of index blocks found in the cache.
Index Blocks Hit+	The cumulative number of index blocks found in the cache.
Index Blocks Read	The number of index blocks read.
Index Blocks Read+	The cumulative number of index blocks read.
Index Size (MB)	The size of the specified index (in megabytes).
In Packets Discards	The number of inbound packets discarded.
In Packets Discards+	The cumulative number of inbound packets discarded.
In Packets Errors	The number of inbound packets that contain errors.
In Packets Errors+	The cumulative number of inbound packets that contain errors.
Link Bandwidth (Mbit/s)	The speed of the network adapter (in megabits per second).
Load Average - 15 Minute	CPU saturation (in percent) - 15 minute sampling average.
Load Average - 1 Minute	CPU saturation (in percent) - 1 minute sampling average.
Load Average - 5 Minute	CPU saturation (in percent) - 5 minute sampling average.
Load Percentage	CPU saturation in percent.
Number of Prepared Transactions+	The cumulative number of prepared transactions.
Number of WAL Files+	The cumulative number of write-ahead log files.
Out Packets Discards	The number of outbound packets discarded.
Out Packets Discards+	The cumulative number of outbound packets discarded.
Out Packets Errors	The number of outbound packets that contain errors.
Out Packets Errors+	The cumulative number of outbound packets that contain errors.
Packets Received	The number of packets received.
Packets Received+	The cumulative number of packets received.
Packets Sent	The number of packets sent.
Packets Sent+	The cumulative number of packets sent.
Size (MB)	The total size of the disk (in megabytes).
Size of Indexes (MB)	The size of indexes on the specified table (in megabytes).

continues on next page

Table 14 – continued from previous page

Metric name	Description
Space Available (MB)	The current disk space available (in megabytes).
Space Used (MB)	The current disk space used (in megabytes).
Table Size (MB)	The size of the specified table (in megabytes).
Tablespace Size (MB)	The size of the specified tablespace (in megabytes).
Temp Buffers (MB)	The size of temporary buffers (in megabytes).
Toast Blocks Hit	The number of TOAST blocks found in the cache.
Toast Blocks Hit+	The cumulative number of TOAST blocks found in the cache.
Toast Blocks Read	The number of TOAST blocks read.
Toast Blocks Read+	The cumulative number of TOAST blocks read.
Total RAM Memory	The total amount of RAM memory on the system (in megabytes).
Total Swap Memory	The total amount of swap space on the system (in megabytes).
Total Table Size w/Indexes and Toast	The total size of the specified table (including indexes and associated oversized attributes).
Transactions Aborted	The number of aborted transactions.
Transactions Aborted+	The cumulative number of aborted transactions.
Transactions Committed	The number of committed transactions.
Transactions Committed+	The cumulative number of committed transactions.
Tuples Deleted	The number of tuples deleted from the specified table.
Tuples Deleted+	The cumulative number of tuples deleted from the specified table.
Tuples Estimated by ANALYZE	The number of visible tuples in the specified table.
Tuples Estimated by ANALYZE+	The cumulative number of visible tuples in the specified table.
Tuples Fetched	The number of tuples fetched from the specified table.
Tuples Fetched+	The cumulative number of tuples fetched from the specified table.
Tuples HOT Updated	The number of tuples HOT updated. In a HOT update, the new tuple resides in the same block as the original tuple and the tuples share an index entry.
Tuples HOT Updated+	The cumulative number of tuples HOT updated. In a HOT update, the new tuple resides in the same block as the original tuple and the tuples share an index entry.
Tuples Inserted	The number of tuples inserted into the specified table.
Tuples Inserted+	The cumulative number of tuples inserted into the specified table.
Tuples Returned	The number of tuples returned in result sets.
Tuples Returned+	The cumulative number of tuples returned in result sets.
Tuples Updated	The number of tuples updated in the specified table.
Tuples Updated+	The cumulative number of tuples updated in the specified table.
WAL Segment Size (MB)	The segment size of the write-ahead log (in megabytes).

Note

The + following the name of a metric means that the data for the metric is gathered cumulatively. Metrics that aren't followed by the '+' sign are collected as a point-in-time value.

26.6 Audit Manager

You can use the PEM Audit Manager to simplify audit log configuration for EDB Postgres Advanced Server instances. With Audit Manager, you can configure logging attributes such as:

- How often PEM collects log files
- The type of database activities that are included in the log files

- How often to rotate log files and when

Audit logs can include the following activities:

- All connections made to the database instance
- Failed connection attempts
- Disconnections from the database instance
- All queries (SELECT statements)
- All DML statements (INSERT, UPDATE, DELETE)
- All DDL statements (for example, CREATE, DROP, ALTER)

Once the audit logs are stored on the PEM server, you can use the Audit Log dashboard to review the information in an easy-to-read form. The Audit Log dashboard allows you to filter the log file by:

- Timestamp range (when an activity occurred)
- The database on which the activity occurred
- The user performing the activity
- The type of command being invoked

26.6.1 Setting the EDB Postgres Advanced Server instance service ID

To configure logging for an EDB Postgres Advanced Server instance, the server must be a PEM-managed server with a bound agent, and the server registration must include the name of a service script. When registering a new server, include the service name in the **Service ID** field on the **Advanced** tab of the New Server dialog box.

Before adding a service name to an existing, registered and connected server, you must disconnect the server:

1. Right-click the server name, and select **Disconnect server** from the context menu.
2. Right-click the server name and select **Properties** from the context menu.
3. Select the **Advanced** tab, and add a service name to the **Service ID** field.

The Service ID field allows the PEM server to stop and start the service.

- The name of the Advanced Server 12 service script is `edb-as-12` .
- The name of the Advanced Server 11 service script is `edb-as-11` .
- The name of the Advanced Server 10 service script is `edb-as-10` .

26.6.2 Setting the EDB Audit Configuration probe

Before configuring audit logging of EDB Postgres Advanced Server servers, you must ensure that the EDB Audit Configuration probe is enabled. To open the **Manage Probes** tab and check the status of the probe, right-click the name of a registered EDB Postgres Advanced Server server in the tree, and select **Management > Manage Probes**.

Ensure that the Enabled column in the Probe Configuration dialog box is set to **Yes** for the EDB Audit Configuration probe. If EDB Audit Configuration isn't enabled, use the **Enabled?** switch on the **Manage Probes** tab to enable it.

26.6.3 Configuring audit logging with the Audit Manager

1. To open the Audit Manager wizard, select **Management > Audit Manager**. The Audit Manager - Welcome dialog box opens. Select **Next** to continue.

2. Use the Select Servers tree to specify the servers to which to apply the auditing configuration. To make a server available in the tree, you must provide the Service ID on the **Advanced** tab of the Create Server dialog box when registering a server for monitoring by PEM. Only EDB Postgres Advanced Server supports auditing. PostgreSQL servers don't appear in the tree.

Select **Next** to continue.

1. The Auditing Parameters Configuration dialog box lets you enable or disable auditing and choose how often log records are collected into PEM. Use the Auditing Parameters Configuration dialog box to specify auditing preferences:

- Use the **Auditing** switch to enable or disable auditing on the specified servers.
- Use the **Audit destination** list to select a destination for the audit logs. Select **File** or **Syslog**. This feature is supported only on EDB Postgres Advanced Server 10 and later releases.
- Use the **Import logs to PEM** switch to periodically import log records from each server to the PEM Server. Set the switch to **Yes** to import log files. The default is **No**.
- Use the **Collection frequency** list to specify how often PEM collects log records from monitored servers when log collection is enabled.
- Use the **Log format** list to select the raw log format to write on each server. If log collection is enabled, the PEM server uses CSV format.
- Use the **File name** field to specify the format used when generating log file names. By default, the format is set to `audit-%Y-%m-%d_%H%M%S` where:

`audit` is the file name specified in the Audit Directory Name field `Y` is the year that the log was stored `m` is the month that the log was stored `d` is the day that the log was stored `H` is the hour that the log was stored `M` is the minute that the log was stored `S` is the second that the log was stored

- Select **Change Log Directory for selected servers?** and use the **Audit Directory Name** field to specify a directory name to which to write the audit logs. The directory resides beneath the data directory on the PEM server.
- Use the **Log directory** box to specify information about the directory in which the log files are saved:

```
-   Move the **Change log directory for selected servers?** switch to **Yes** to␣
↪enable the **Directory name** field.

-   Use the **Directory name** field to specify the name of the directory on each␣
↪server to write audit logs to. The directory specified is created as a subdirectory␣
↪of the data directory on the server.
```

Select **Next** to continue.

1. The Audit log configuration dialog box is available only if you enabled auditing on the Auditing Parameters Configuration dialog box. Use the Audit Log Configuration dialog box to specify log configuration details to apply to each server:

- Use the **Connection attempts** switch to specify whether to log connection attempts. Specify:

None to disable connection logging

All to indicate that all connection attempts are logged

Failed to log any connection attempts that fail

- Use the **Disconnection attempts** switch to specify whether to log disconnections. Specify:

None if you don't want to log disconnections

All to enable disconnection logging

- Use the **Log statements** field to specify the statement types that are logged. Select from:

Select _ Log all statements that include the `SELECT` keyword.

Error _ Log all statements that result in an error.

DML _ Log all DML (data modification anguage) statements.

DDL _ Log all DDL (data definition language) statements, that is, those that add, delete or alter data. Select **Select All** to select all statement types.

Select **Unselect All** to clear all statement types.

- Use the **Audit tag** field to specify a tracking tag for the collected logs. Audit tagging functionality is available only for supported versions of EDB Postgres Advanced Server.
- Use the **Log rotation** box to specify how to manage the log files on each server:

Use the **Enable?** switch to rotate log files. Use a new log file periodically to prevent a single file from becoming too large.

Use the **Day** list to select days on which to rotate the log file.

Use the **Size (MB)** field to specify a size in megabytes at which to rotate the log file.

Use the **Time (seconds)** field to specify the number of seconds between log file rotations.

Select **Next** to continue:

1. Use the Schedule Auditing Changes dialog box to determine when auditing configuration changes take effect.

- Select **Configure logging now?** if you want the auditing configuration changes to take place immediately. The affected database servers restart so the auditing changes can take effect.
- Use the **Time?** selector to schedule the auditing configuration changes to take place at some point in the future. Select the desired date and time from the lists. The affected database servers restart at the specified date and time to put the auditing changes into effect.

Select **Finish** to complete the auditing configuration process.

Audit Manager schedules a job to apply the configuration to each server. The job consists of two tasks: one to update the audit logging configuration on the server and one to restart the server with the new configuration.

You can use the **Scheduled Tasks** tab to review a list of scheduled jobs. To open the **Scheduled Tasks** tab, select the name of a server or agent and select **Management > Scheduled Tasks**.

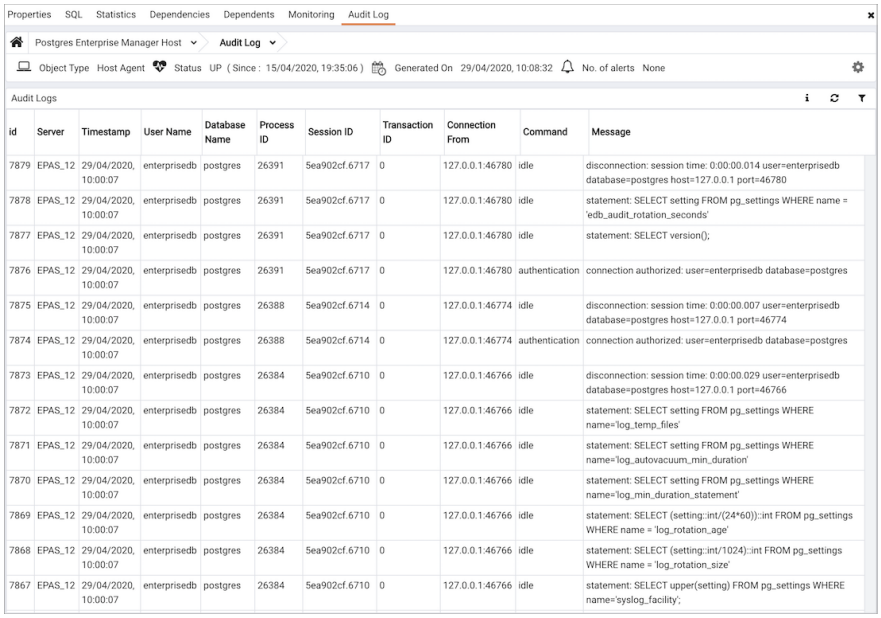
26.6.4 Viewing the log with the Audit Log dashboard

Use the Audit Log dashboard to view the audit log from EDB Postgres Advanced Server database instances.

To open the Audit Log dashboard, right-click a server or agent node and select **Audit Log Analysis** from the **Dashboards** menu. The Audit Log dashboard displays the audit records in reverse chronological order.

You can use filtering to limit the number of audit records that are displayed. Select **Show Filters** to expose the filters panel. Use the filters panel to provide selection criteria for the audit records you want to display.

- Use the **Start** field to specify a start date for the report.
- Use the **End** field to specify an end date for the report.
- Use the **User** field to display only those entries where the activity was started by the given Postgres user.
- Use the **Database** field to display only those entries where the activity was issued on the given database.



id	Server	Timestamp	User Name	Database Name	Process ID	Session ID	Transaction ID	Connection From	Command	Message
7879	EPAS_12	29/04/2020, 10:00:07	enterprisedb	postgres	26391	5ea902cf.6717	0	127.0.0.1:46780	idle	disconnection: session time: 0:00:00.014 user=enterprisedb database=postgres host=127.0.0.1 port=46780
7878	EPAS_12	29/04/2020, 10:00:07	enterprisedb	postgres	26391	5ea902cf.6717	0	127.0.0.1:46780	idle	statement: SELECT setting FROM pg_settings WHERE name = 'edb_audit_rotation_seconds'
7877	EPAS_12	29/04/2020, 10:00:07	enterprisedb	postgres	26391	5ea902cf.6717	0	127.0.0.1:46780	idle	statement: SELECT version();
7876	EPAS_12	29/04/2020, 10:00:07	enterprisedb	postgres	26391	5ea902cf.6717	0	127.0.0.1:46780	authentication	connection authorized: user=enterprisedb database=postgres
7875	EPAS_12	29/04/2020, 10:00:07	enterprisedb	postgres	26388	5ea902cf.6714	0	127.0.0.1:46774	idle	disconnection: session time: 0:00:00.007 user=enterprisedb database=postgres host=127.0.0.1 port=46774
7874	EPAS_12	29/04/2020, 10:00:07	enterprisedb	postgres	26388	5ea902cf.6714	0	127.0.0.1:46774	authentication	connection authorized: user=enterprisedb database=postgres
7873	EPAS_12	29/04/2020, 10:00:07	enterprisedb	postgres	26384	5ea902cf.6710	0	127.0.0.1:46766	idle	disconnection: session time: 0:00:00.029 user=enterprisedb database=postgres host=127.0.0.1 port=46766
7872	EPAS_12	29/04/2020, 10:00:07	enterprisedb	postgres	26384	5ea902cf.6710	0	127.0.0.1:46766	idle	statement: SELECT setting FROM pg_settings WHERE name='log_temp_files'
7871	EPAS_12	29/04/2020, 10:00:07	enterprisedb	postgres	26384	5ea902cf.6710	0	127.0.0.1:46766	idle	statement: SELECT setting FROM pg_settings WHERE name='log_autovacuum_min_duration'
7870	EPAS_12	29/04/2020, 10:00:07	enterprisedb	postgres	26384	5ea902cf.6710	0	127.0.0.1:46766	idle	statement: SELECT setting FROM pg_settings WHERE name='log_min_duration_statement'
7869	EPAS_12	29/04/2020, 10:00:07	enterprisedb	postgres	26384	5ea902cf.6710	0	127.0.0.1:46766	idle	statement: SELECT (setting::int/(24*60))::int FROM pg_settings WHERE name = 'log_rotation_age'
7868	EPAS_12	29/04/2020, 10:00:07	enterprisedb	postgres	26384	5ea902cf.6710	0	127.0.0.1:46766	idle	statement: SELECT (setting::int/1024)::int FROM pg_settings WHERE name = 'log_rotation_size'
7867	EPAS_12	29/04/2020, 10:00:07	enterprisedb	postgres	26384	5ea902cf.6710	0	127.0.0.1:46766	idle	statement: SELECT upper(setting) FROM pg_settings WHERE name='syslog_facility';

Fig. 9: The Audit Log dashboard

- Use the **Command type** field to display only those entries where the activity was of the given type. Command types you can specify are **idle**, **authentication**, and **SELECT**. (For viewing SQL statements from user applications, specify the idle command type.)

Select **Filter** to apply the filtering criteria to the log entries.

26.7 Log Manager

You can use the PEM Log Manager to simplify server log configuration for Postgres instances. With Log Manager, you can modify all of your server log parameters including:

- Where log files are written
- How often log files are written
- The type of information written to log files
- The format of log file entries
- Log rotation properties

Before using Log Manager to define logging properties for a server, you must specify the name of the associated Advanced Server or PostgreSQL database server in the **Service ID** field on the **Advanced** tab of the New Server Registration or Properties dialog box. The server is available for configuration on the Server Selection dialog box only if you specify the name of the service in the **Service ID** field.

For example, suppose you're setting logging preferences for an Advanced Server 9.4 instance that resides on a Linux host. Set the **Service ID** field on the **Advanced** tab of the **Properties** dialog box for the monitored server to `ppas-9.4`.

Note

- Log Manager depends on Settings and Server Log Configuration probes to populate all the fields in the wizard. Therefore, ensure that those probes for selected servers are enabled. In addition, set the execution frequency for those probes to a minimum to ensure that Log Manager reflects the latest log configurations. - Rerun Log Manager if you make any manual changes related to logging in the configuration files, such as *postgresql.conf* . PEM doesn't reflect those changes automatically.

To configure logging for a Postgres instance, you must register the server as a PEM-managed server, and the registration information must include the name of a service script.

1. To open Log Manager, in the PEM client, select **Management > Log Manager**. The wizard opens and welcome message appears. Select **Next**.
2. The Server Selection dialog box displays a list of the server connections monitored by PEM. Select the check boxes next to the names of servers to which you want the Log Manager wizard to apply the specified configuration. Log Manager is disabled for any server displaying a red exclamation mark to the left of its name in the Server selection tree. A server might not be enabled for several reasons:

- Log Manager can configure only a server that specifies a service ID on the **Advanced** tab of the Properties dialog box. To provide a service ID: 1. In the tree, right-click the server name and select **Disconnect Server** from the context menu. 1. If prompted, provide a password. 1. From the context menu for the server, select **Properties**. 1. On the **Advanced** tab, enter the name of the service in the **Service ID** field. 1. Select **Save**.
- If the PEM agent bound to the server doesn't have enough privileges to restart the server, the server is disabled.
- If the PEM agent bound to the server is an older version than the associated PEM server, the server is disabled.

Select **Next**.

3. Use the Log configuration dialog box to specify how often to import log files to PEM and to specify log rotation details.

Options in the **Import Logs** box specify how often to import the log files to PEM:

- Use the switch next to **Import logs to PEM** to specify whether to import the log files to PEM and to display them on the Server Log Analysis dashboard.
- Use the **Import Frequency** list to specify how often to import log files to PEM.

Use the **Log rotation configuration** box to specify the maximum length (lifespan or size) of a log file:

- Use the **Rotation Size** field to specify the maximum size of a log file in megabytes. The default value is 10 MB. Set to 0 if you don't want to place a limit on the maximum size of a log file.
- Use the **Rotation Time** field to specify the number of whole days to store in each log file. The default value is 1 day.

Use the **Truncation on Rotation** switch to specify server behavior for time-based log file rotation:

- Select **On** to overwrite any existing log file that has the same name as a new file.
- Select **Off** to append any new log file entries to an existing log file with the same name as a new log file. This is the default behavior.

Select **Next**.

4. Use the Where to Log dialog box to specify where to write log files:

- Select an option from the **Log Destination** box to specify a destination for the server log output: - Set the **stderr** switch to **Yes** to write log files to stderr. - Set the **csvlog** switch to **Yes** to write log files in a comma-separated value format. This option is enabled and not editable if you select **Import logs to PEM** on the Schedule dialog box. If you aren't importing server log files to PEM, you can edit this option. - Set the **syslog** switch to **Yes** to

write log files to the system log files. - On Windows, set the **eventlog** switch to **Yes** to write log files to the event log.

- Use the **Log collection** box to specify your collection preferences: - Set the **Log Collector** switch to **Enable** to redirect captured log messages (directed to stderr) into log files. - Set the **Log Silent Mode** switch to **Enable** to run the server silently in the background, disassociated from the controlling terminal.
- Use the **Log Directory** box to specify log file location preferences: - Set the **Change log directory for selected servers?** switch to **Yes** to maintain each set of log files in a separate directory. - Use the **Directory name** field to specify the directory to which to write log files. The directory resides beneath the `pg_log` directory under the installation directory of the monitored server.
- Use the **Log File Name** field to specify a format for the log file name. If set to **Default**, the format is `enterprisedb-%Y-%m-%d_%H%M%S` , where: - `enterprisedb` is the file name prefix - `Y` is the year that the log was stored - `m` is the month that the log was store - `d` is the day that the log was stored - `H` is the hour that the log was stored - `M` is the minute that the log was store - `S` is the second that the log was stored
- When logging to syslog is enabled: - Use the **Syslog Facility** list to specify the syslog facility to use. - Use the **Syslog Ident** field to specify the program name that identifies EDB Postgres Advanced Server entries in system logs.

Select **Next**.

5. Use the When to Log dialog box to specify the events that initiate a log file entry.

- The severity levels in order of severity, from most severe to least severe, are:
  - **panic** _ Errors that cause all database sessions to abort.
  - **fatal** _ Errors that cause a session to abort.
  - **log** _ Information messages of interest to administrators.
  - **error** _ Errors that cause a command to abort.
  - **warning** _ Error conditions in which a command completes but might not perform as expected.
  - **notice** _ Items of interest to users. This is the default.
  - **info** _ Information implicitly requested by the user.
  - **debug5** through **debug1** _ Detailed debugging information useful to developers.
- Use the **Client min messages** list to specify the lowest severity level of message sent to the client application.
- Use the **Log min messages** list to specify the lowest severity level to write to the server log.
- By default, when an error message is written to the server log, the text of the SQL statement that initiated the log entry isn't included. Use the **Log min error statement** list to specify a severity level that triggers SQL statement logging. If a message is of the specified severity or higher, the SQL statement that produced the message is written to the server log.
- Use the **Log min duration statement** list to specify a statement duration, in milliseconds. Any statements that exceed the specified number of milliseconds are written to the server log. A value of **-1** disables all duration-based logging. A value of **0** logs all statements and their duration.
- Use the **Log temp files** field to specify a file size, in kilobytes. When a temporary file reaches the specified size, it's logged. Specify a value of **-1** (the default) to disable this functionality.
- Use the **Log autoVacuum min duration** field to specify a time length, in milliseconds. If auto-vacuuming exceeds the length of time specified, the activity is logged. Specify a value of **-1** (the default) to disable this functionality.

Select **Next**.

6. Use the What to Log dialog box to specify log entry options that are useful for debugging and auditing.

Use the switches in the **Debug options** box to include information in the log files related to query execution that might be of interest to a developer:

- Set the **Parse tree** switch to **Yes** to include the parse tree in the log file.
- Set the **Rewriter output** switch to **Yes** to include query rewriter output in the log file.
- Set the **Execution plan** switch to **Yes** to include the execution plan for each executed query in the log file.

When the **Indent Debug Options Output in Log** switch is set to **Yes**, the server indents each line that contains a parse tree entry, a query rewriter entry, or query execution plan entry. While indentation makes the resulting log file more readable, it results in a longer log file.

Use the switches in the **General Options** box to include auditing information in the log file:

- Set the **Checkpoints** switch to **Yes** to include checkpoints and restartpoints in the server log.
- Set the **Connections** switch to **Yes** to include each attempted connection to the server as well as successfully authenticated connections in the server log.
- Set the **Disconnections** switch to **Yes** to include a server log entry for each terminated session that provides the session information and session duration.
- Set the **Duration** switch to **Yes** to include the amount of time required to execute each logged statement in the server log.
- Set the **Hostname** switch to **Yes** to include both the IP address and host name in each server log entry. By default, only the IP address is logged. This might cause a performance penalty.
- Set the **Lock Waits** switch to **Yes** to write a log entry for any session that waits longer than the time specified in the `deadlock_timeout` parameter to acquire a lock. This is useful when trying to determine if lock waits are the cause of poor performance.

Use the **Error verbosity** list to specify the detail written to each entry in the server log:

- Select **default** to include the error message, DETAIL, HINT, QUERY, and CONTEXT in each server log entry.
- Select **terse** to log only the error message.
- Select **verbose** to include the error message, the DETAIL, HINT, QUERY, and CONTEXT error information, SQLSTATE error code and source code file name, the function name, and the line number that generated the error.

Use the **Prefix string** field to specify a printf-style string that is written at the beginning of each log file entry. For information about the options supported, see the log_line_prefix documentation in the Postgres core documentation .

Use the **Statements** list to specify the SQL statements to include in the server log:

- Specify **none** (the default) to disable logging of SQL statements.
- Specify **ddl** to instruct to log ddl (data definition language) statements, such as CREATE, ALTER, and DROP.
- Specify **mod** to log all ddl statements as well as all dml (data modification language) statements, such as INSERT, UPDATE, DELETE, TRUNCATE, and COPY FROM.
- Specify **all** to log all SQL statements.

Select **Next**.

7. Use the Schedule Logging Changes dialog box to specify when logging applies configuration changes:

- Set the **Configure logging now** switch to **Yes** to enable your configuration preferences. The server restarts when you complete the Log Manager wizard.

- Set **Configure logging now** to **No** to use the **Schedule it for some other time** calendar selector to specify a convenient time to apply logging configuration preferences and for the server to restart.

When you apply the configuration changes specified by the Log Manager wizard, the server restarts, temporarily interrupting use of the database server for users.

8. Select **Finish** to exit the wizard. Either restart the server or schedule the server restart for the time specified on the scheduling dialog box.

26.7.1 Reviewing the Server Log Analysis dashboard

After invoking the Log Manager wizard and importing your log files to PEM, you can use the Server Log Analysis dashboard to review the log files for a selected server. To open the Server Log Analysis dashboard, right-click the name of a monitored server in the PEM client tree and select **Dashboards > Server Log Analysis**.

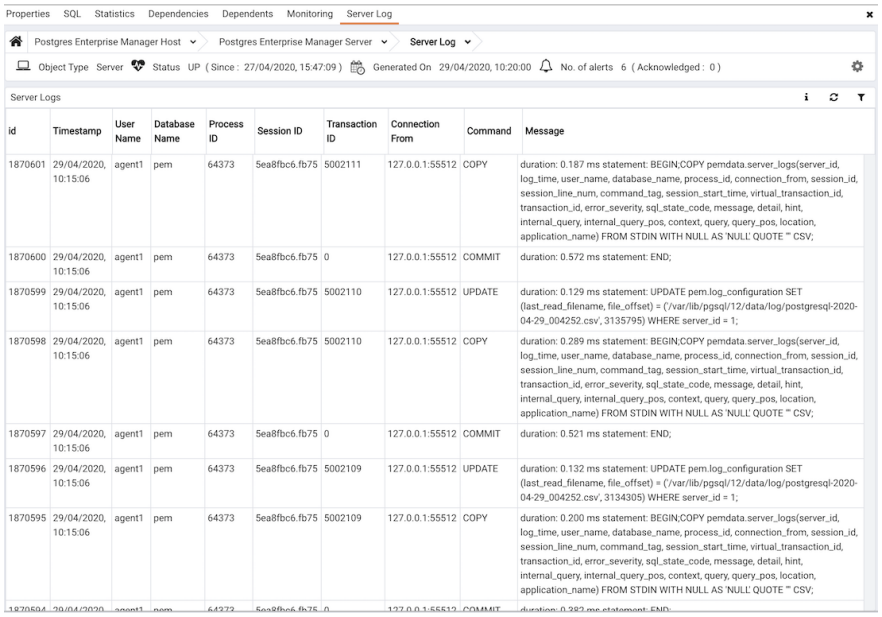


Fig. 10: The Server Log Analysis dashboard

The header information on the Server Log Analysis dashboard displays the date and time that the server was started, the date and time that the page was last updated, and the current number of triggered alerts.

Entries in the Server Log table appear in chronological order, with the most recent log entries first. Use the scroll bars to navigate through the log entries or to view columns that are off of the display.

Headings at the top of the server log table identify the information stored in each column. Hover over a column heading to view a tooltip that contains a description of the content of each column.

You can use filtering to limit the number of server log records that are displayed. Select **Show Filters** to open the filters panel and define a filter. Use the **filter definition** box to describe the selection criteria for selecting a subset of a report to display:

- Use the **From** field to specify a starting date for the displayed server log.
- Use the **To** field to specify an ending date for the displayed server log.
- Enter a role name in the **Username** field to display transactions performed only by that user.
- Enter a database name in the **Database** field to limit the displayed records to transactions that were performed against the specified database.

- Use the **Command Type** field to specify selection criteria for the commands that appear in the filtered report.

After you specify the criteria for filtering the server logs, select **Filter** to display the filtered server log in the Server Log table.

26.8 Charts

You can use the **Manage Charts** tab to create or modify a custom line chart or table or to import a Capacity Manager template to use in a custom chart. After defining a chart, you can display the chart on a custom dashboard.

To open the **Manage Charts** tab, in the PEM client, select **Management > Manage Charts**. The **Manage Charts** tab provides a Quick Links menu for accessing dialog boxes to:

- Create a new chart for use on a custom dashboard.
- Import a Capacity Manager template to use as a template for creating a custom chart.

The Custom Charts table displays a list of charts you defined. When a chart is new, the font is green. When you add a chart or refresh the screen, the font is black. Use the search box in the upper-right of the Custom Charts table to search through your custom charts. Specify:

- Chart name
- Type
- Level
- Metrics category

Use icons to the left of a chart name in the Custom Charts table to manage a chart:

- Select **Edit** to open the Chart Configuration wizard and modify aspects of the chart or table.
- Select **Delete** to delete the selected chart.

26.8.1 Creating a custom chart

Select **Create New Chart** in the Quick Links section of the **Manage Charts** tab to open the Create Chart wizard. The wizard takes you through the steps required to define a new chart.

1. Use the Configure Chart dialog box to specify general information about the chart:

- Specify the name of the chart in the **Name** field.
- Use the list in the **Category** field to specify the category in which to display this chart. When adding a custom chart to a custom dashboard, you can select the chart from the category specified.
- Use the **Type** field to specify if the chart is a line chart or a table.
- Provide a description of the chart in the **Description** field. The description is displayed on a custom dashboard when the user selects the information icon next to the chart.

After you complete the fields on the Configure Chart dialog box, select **Next**.

1. Use the Select Metrics dialog box to select the metrics to display on the chart.

- Use the **Metric level** list to specify the level of the PEM hierarchy from which you want to select metrics. You can specify **Agent**, **Database**, or **Server**. Each level offers access to a unique set of probes and metrics.
- Use the tree in the **Available metrics** box to select the metrics to display on the chart.

If you're creating a table, you can select metrics from only one probe. Each node of the tree lists the metrics returned by a single probe. Expand a node of the tree and select the check box next to a metric name to include that metric data in the table.

If you're creating a line chart, expand the nodes of the tree and double-click each metric that you want to include in the chart.

- Use the Selected Metrics panel to specify how the metric data appears in your chart. The selection panel displays the name of the metric in the nonmodifiable Metric [Probe] column. You can:

```
-    Select the trash can to delete a metric from the list of selected metrics.
-    Use the lists in the Selection Criteria column to specify the order of the data␣
↪displayed.
-    Use the **Limit** field to specify the number of rows in a table or lines in a␣
↪chart: up to 32 lines and 100 rows.
```

- If you're creating a line chart, PEM supports comparisons of cross-hierarchy metrics.

```
-    Select **Compare** to select one or more probe-specific attributes (such as CPUs,
↪ interfaces, and databases) to compare in the chart.
-    Select **Copy** to apply your selections to all of the metrics for the same␣
↪probe.
```

After you complete the fields on the Select Metrics dialog box, select **Next**.

1. Use the Set Options dialog box to specify display options for your chart:

- Use the **Auto Refresh** field to specify the number of minutes between chart updates. Choose a value from 1 to 120. The default auto refresh rate is 2 minutes.
- Use fields under **Line chart options** to specify display preferences for a line chart:

```
-    Use the **Points to plot** field to specify the maximum number of points to plot␣
↪on the chart.
-    Use the fields to the right of **Historical span** to specify how much␣
↪historical data to display on the chart in days, hours, and minutes.
```

- Use the fields in the **Data extrapolation** box to specify whether PEM generates extrapolated data based on historical data:

```
-    Select **No Extrapolation** to omit extrapolated data from the chart.
-    Select **Span** to use the **Days** and **Hours** selectors to specify the␣
↪period of time spanned by the metrics on the chart.
-    Select **Threshold** to use threshold selectors to specify a maximum or minimum␣
↪value for the chart.
```

After you complete the fields in the Set Options dialog box, select **Next**.

1. Use the Set Permissions dialog box to specify display options for your chart.

- Set the **Share with all** slider to **Yes** to make the chart available to all authorized users. Select **No** to restrict access to the users or groups specified in the **Access permissions** field.
- Use the **Access permissions** field to select the groups you want to give access to the chart to.

After you finish defining the chart, select **Finish** to save your edits and add your chart to the list on the **Manage Charts** tab.

26.8.2 Importing a Capacity Manager template

Select **Import Capacity Manager Template** in the Quick Links section of the **Manage Charts** tab to open the Create Chart dialog box and use a Capacity Manager template as a starting point for a chart or table.

1. When the Create Chart dialog box opens, provide information about the custom chart:

- Use the **Import capacity template** list to select the name of the template on which the chart is based.
- Specify the name of the chart in the **Name** field.
- Use the **Category** list to specify the category in which to display this chart. When adding a custom chart to a custom dashboard, you can select the chart from this category.
- Use the **Type** field to specify if the chart is a line chart or a table.
- Provide a description of the chart in the **Description** field. The description is displayed on the custom dashboard when user selects the information icon next to the chart.

Select **Next**.

1. The Select Metrics window allows you to review the metrics specified by the selected template. The bottom panel of the chart editor displays the metrics to include in the chart. You can't modify the metrics included in the chart using the chart editor. To modify the metrics, use the Capacity Manager utility to update the template.

After you review the metrics, select **Next**.

1. Use the Set Options window to specify display options for your chart:

- Use the **Auto Refresh** field to specify the number of minutes between chart updates. Choose a value from 1 to 120. The default auto refresh rate is 2 minutes.
- Use the **Data extrapolation** box to specify the time period covered by the chart. You can either:
- Select **Historical days and extrapolated days** and provide: - The number of days of historical data to chart in the **Historical** field. - The number of projected days to chart in the **Extrapolated** field.
- Select **Historical days and threshold** and provide: - The number of days of historical data to chart in the **Historical** field. - The threshold value at which the chart ends.

After you complete the Set Options window, select **Next**.

1. Use the Set Permissions window to specify display options for your chart:

    - Set the **Share with all slider** to **Yes** to make the chart available to all authorized users. Set it to **No** to restrict access to the users or groups specified in the **Access permissions** field. - Use the **Access permissions** field to select the groups to give access to the chart to.

After you finish defining the chart, select **Finish** to save your edits and add your chart to the list on the **Manage Charts** tab.

26.9 Dashboards

PEM displays performance statistics through a number of dashboards. Each dashboard contains a series of summary views that contain charts, graphs, and tables that display the statistics related to the selected object.

26.9.1 Dashboards overview

The PEM client displays the Global Overview dashboard when it connects to the PEM server. Additional dashboards provide statistical information about monitored objects.

Opening dashboards

Dashboards are presented in a hierarchy comparable to the PEM client tree control. The dashboard for each object in the tree control displays the information for that object as well as for any monitored object that resides below that level in the tree control, if appropriate.

Each dashboard header displays the date and time that the server was started, if relevant, the date and time that the dashboard was last updated, and the current number of triggered alerts. Navigation menus displayed in the dashboard header provide easy access to other dashboards. Menus are organized hierarchically. Only those menus appropriate for the object currently highlighted in the tree control are available.

- Select **Global Overview** from any dashboard to return to the Global Overview dashboard.
- Select the name of an agent from the **Agents** menu to navigate to the Operating System Analysis dashboard for that agent.
- Select a server name from the **Servers** menu to navigate to the Server Analysis dashboard for that server.
- Select a database name from the **Databases** menu to navigate to the Database Analysis dashboard for that database.
- Use the **Dashboards** menu to navigate to informational dashboards at the global level or for the selected agent, server, or database.

Dashboards display statistical information in the form of:

- Tables that provide statistical information collected by a PEM probe.
- Pie charts that display information collected by the most recent execution of a probe.
- Bar graphs that display comparative statistics collected by the most recent execution of a probe.
- Line graphs that display statistical data collected by PEM probes.

You can open a dashboard using either of these techniques:

- from the **Management > Dashboards** menu, select an active dashboard name.
- Right-click the name of a monitored object in the tree. From the **Dashboards** menu, select the name of the dashboard to review.

Each dashboard is displayed on the **Monitoring** tab in the main panel of the client window. After opening a dashboard, you can navigate to other dashboards in the same tab.

Each dashboard header includes navigation menus that allow you to navigate to other dashboards. Use your browser's forward and back buttons to scroll through previously viewed dashboards. Use **Refresh** to update the current dashboard.

To sort statistics that are provided in table form, select a column heading. Select it again to reverse the sort order. Each table offers a stable sort feature. For example, to sort a table by ascending Session ID in each user name group, sort first by the Session ID column, then sort by the User Name column.

Hover over the upper-right corner of each graph, chart, or table to reveal the PEM client toolbar icons. Hover over an icon to display a tooltip that briefly explains the icon's functionality:

- Use the **Refresh** icon to update the information displayed on a dashboard.
- Use the **Save Chart as Image** icon to save the selected chart as a `.jpeg` image.
- Use the **Full Screen** icon to enlarge the chart to reveal granular details about the charted data.
- Use the **Personalize the chart configuration** icon to access a control panel that allows you to select chart-specific display details.
- Hover over the **Explain** icon to review a description of the information shared in the graph or chart.

In the lower-right corner of each graph or chart is a legend that identifies each item plotted in the graph or chart.

If it's displayed, select the information icon in the upper-left corner of a chart to display a note about the chart content and, if applicable, a link that allows you to enable one or more probes that retrieve content for the chart.

26.9.2 Available dashboards

PEM offers the following dashboards.

Alerts dashboard

The Alerts dashboard displays the currently triggered alerts. If opened from the Global Overview, the dashboard displays the current alerts for all monitored nodes on the system. If opened from a node in a server, the report shows alerts related to that node and all monitored objects that reside below that object in the tree.

Audit Log Analysis dashboard

For EDB Postgres Advanced Server users, the Audit Log Analysis dashboard allows you to browse the audit logs that were collected from instances with audit logging and collection enabled.

PGD Admin dashboard

The PGD Admin dashboard displays overview information about the EDB Postgres Distributed node, group, and worker.

PGD Group Monitoring dashboard

The PGD Group Monitoring dashboard displays information about EDB Postgres Distributed group subscription and group replication slots.

PGD Node Monitoring dashboard

The PGD Node Monitoring dashboard displays information about EDB Postgres Distributed node slots, node replication rates, and conflict history summary for the selected node.

Database Analysis dashboard

The Database Analysis dashboard displays performance statistics for the selected database.

I/O Analysis dashboard

The I/O Analysis dashboard displays I/O activity across various areas such as object DML activity and log operations.

Memory Analysis dashboard

The Memory Analysis dashboard supplies statistics concerning various memory-related metrics for the Postgres server.

Object Activity Analysis dashboard

The Object Activity Analysis dashboard provides performance details on tables/indexes of a selected database.

Operating System Analysis dashboard

The Operating System Analysis dashboard supplies information regarding the performance of the underlying machine's operating system.

Probe Log Analysis dashboard

The Probe Log Analysis dashboard displays any error messages returned by a PEM agent.

Server Analysis dashboard

The Server Analysis dashboard provides general performance information about the overall operations of a selected Postgres server.

Server Log Analysis dashboard

The Server Log Analysis dashboard allows you to filter and review the contents of server logs that are stored on the PEM server.

Session Activity Analysis dashboard

The Session Activity Analysis dashboard provides information about the session workload and lock activity for the selected server

Session Waits Analysis dashboard

The Session Waits Analysis dashboard provides an overview of the current DRITA wait events for an Advanced Server session.

Storage Analysis dashboard

The Storage Analysis dashboard displays space-related metrics for tablespaces and objects.

System Waits Analysis dashboard

The System Waits Analysis dashboard displays a graphical analysis of system wait information for an EDB Postgres Advanced Server session.

Streaming Replication Analysis dashboard

The Streaming Replication Analysis dashboard displays statistical information about WAL activity for a monitored server and allows you to monitor the status of Failover Manager clusters.

26.9.3 Dashboard configuration

Options on the Dashboard Configuration dialog box allow you to link the timelines of all of the line graphs on the dashboard. To open the Dashboard Configuration dialog box, select **Settings** in the dashboard header.

Use the Dashboard Configuration dialog box to control attributes of the charts displayed on the dashboard:

- Set **Link timelines of all the line charts** to **Enable** to apply the specified timeline to line graphs displayed on the dashboard. If set to **Disable**, your preferences are preserved for later use but don't modify the amount of data displayed.
- Use the **Days** selector to specify the number of days of gathered data to display on line graphs.
- Use the **Hour(s)** selector to specify the number of hours of gathered data to display on line graphs.
- Select **Remember configuration for this dashboard** to apply the customized time span to only the current dashboard only. Leave it cleared to apply the time span globally to line graphs on all dashboards.

Settings specified on the Dashboard Configuration dialog box are applied only to the current user's session.

After you specify your preferences, select **Save**.

26.9.4 Custom dashboards

PEM displays performance statistics on a number of system-defined dashboards. Each dashboard contains a series of summary views that contain charts, graphs, and tables that display statistics related to the selected object. You can also create custom dashboards with your choice of charts and layout.

To create a custom dashboard, first open an existing dashboard at the same level as the one you wish to create. For example, if you wish to create a new server-level dashboard, open any of the existing server-level dashboards.

Next, click on the . . . menu at the top right of the dashboard and select **Create Dashboard**. This opens a visual editor with which you can design the dashboard.

Click **Save** when you are done and the dashboard will appear in the list of dashboards shown in the breadcrumb bar at the top of the screen.

To modify an existing dashboard, select **Edit** to reopen the visual editor.

To delete a dashboard, select **Delete** next to the dashboard name.

To change the permissions of an existing dashboard, select **Share Dashboard**.

To choose who can see this dashboard, click the cog icon next to the dashboard name. By default dashboards are shared with all users. You can make the dashboard private, or restrict it to certain roles.

26.10 Remote monitoring

Remote monitoring is monitoring your Postgres cluster using a PEM Agent residing on a different host.

To remotely monitor a Postgres cluster with PEM, you must register the cluster with PEM and bind a PEM agent. See *Registering a Postgres server* for more information.

The following scenarios require remote monitoring using PEM:

- Postgres cluster running on AWS RDS
- Postgres cluster running on BigAnimal

PEM remote monitoring supports:

Feature Name	Remote monitoring supported?	Comments
Query information	Yes	
Viewing system reports	Yes	
Query information	Limited	There's no correlation between the Postgres cluster and operating system metrics.
Query information	Limited	When you run an alert script on the Postges cluster, it runs on the machine where the bound PEM agent is running and not on the actual Postgres cluster machine.
Query information	Limited	Some dashboards might not be able to show complete data. For example, the operating system information where the Postgres cluster is running isn't displayed as it isn't available.
Manage probes	Limited	Some of the PEM probes don't return information, and some of the functionality might be affected. For details about probe functionality, see *Query information* .
Query information	Limited	Scheduled tasks work only for Postgres clusters, and scripts run on a remote agent.

continues on next page

Table 15 – continued from previous page

Feature Name	Remote monitoring supported?	Comments
Query information	Limited	The Core Usage reports don't show complete information. For example, the platform, number of cores, and total RAM aren't displayed.
Query information	No	
Query information	No	

CHAPTER 27

Monitoring Barman

Barman (Backup and Recovery Manager) is an open-source administration tool for remote backups and disaster recovery of PostgreSQL servers in business-critical environments. It relies on PostgreSQL's point-in-time recovery technology, allowing DBAs to remotely manage a complete catalog of backups and the recovery phase of multiple remote servers from one location. For more information, see *Monitoring Barman* .

You can monitor a Barman server through the PEM console.

27.1 Prerequisites for monitoring Barman

Before adding a Barman server to the PEM console:

- You must manually install and configure Barman on the Barman host. For more information about installing and configuring Barman, see *Monitoring Barman* .
- Install the pg-backup-api tool on Barman host. For more information about installing, see pg-backup-api .

27.2 Configuring a Barman server

You can configure and edit your Barman server using:

- PEM web client
- `pemworker` command line

27.2.1 Using PEM web client

27.2.2 Configure

You can use the Create > Barman Server dialog box to register an existing Barman server with the PEM server. To open the dialog box, right-click the Barman Servers node and select **Create > Barman Server**.

Use the **General** tab to describe the general properties of the Barman server:

- Use the **Name** field to specify a name for the server. The name identifies the server in the browser tree.

- Use the **URL** field to specify the URL of the host where Barman is installed.
- Use the **Team** field to specify a PostgreSQL role name. Only PEM users who are members of this role, who created the server initially, or have superuser privileges on the PEM server can see this server when they log on to PEM. If this field is left blank, all PEM users see the server.

Use the **PEM Agent** tab to specify connection details for the PEM Agent:

- Use the **Bound Agent** field to select the agent that you want to configure as a Barman server. Only those PEM agents that are supported for Barman are listed.
- Use the **Probe Frequency** field to specify the number of seconds to execute the probes with the specified interval.
- Use the **Heartbeat** field to specify the interval to check the availability of PEM agent in seconds.

Note

After registering the Barman server, you need to restart the PEM agent.

27.2.3 Editing

To edit your Barman server, right-click the server from the browser tree and select **Properties**.

- Use the **PEM Agent** tab to modify the bound agent, probe frequency, and heartbeat. Only the owner of the Barman server can modify the fields on the **PEM Agent** tab.
- Use the **Information** tab to view the detailed information about your Barman server. This tab gets populated whenever the Barman related probes are executed.
- Use the **Configuration** tab to view the configuration settings of your Barman server. This tab gets populated whenever the Barman related probes are executed.

27.2.4 Using pemworker command line

You can configure Barman server using `pemworker` command line options.

```
$ ./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-
```

(continues on next page)

(continued from previous page)

```
↪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.)
```

Note

After registering the Barman server, you need to restart the PEM agent.

27.3 Viewing the Barman server details on a PEM dashboard

Once the Barman server is configured, you can see the entire backup- and server-related details for that Barman server on the PEM dashboard.

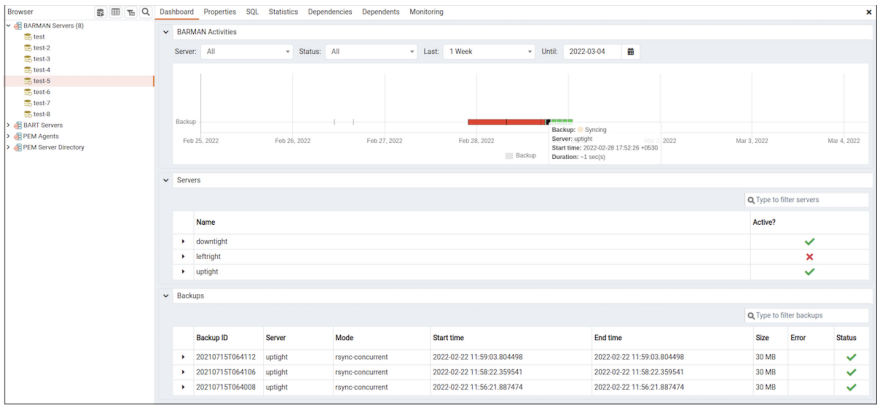


Fig. 1: Barman dashboard

When you select a monitored Barman server, details of all the associated database servers along with their activities are displayed as a chart on the dashboard in the Barman Activities panel. You can select the activities on any criteria that you specify in the filter boxes (the database server, status, duration, or date).

The Servers panel displays a list of all the database servers managed by that Barman server along with the active status.

The Backups panel displays a list of all the database server backups managed by that Barman server. You can filter the list to display the details of any database server. You can also filter the list on any criteria that you specify in the filter box. Typically, this filter works with any kind of string value (excluding date, time, and size) listed under the columns. For example, you can enter `tar` to filter the list and display only those backups that are in tar format.

Backup details include the Backup ID, Server, Mode, Start time, End time, Size, Error, and Status column.

CHAPTER 28

Monitoring EDB Postgres Distributed

EDB Postgres Distributed provides multi-master replication and data distribution with advanced conflict management, data-loss protection, and throughput up to 5X faster than native logical replication , and enables distributed PostgreSQL clusters with high availability up to five 9s. Before you monitor nodes in a EDB Postgres Distributed cluster through the PEM console, you must first deploy a EDB Postgres Distributed cluster and ensure that your database nodes are up and running. For more information on installing EDB Postgres Distributed see *Monitoring EDB Postgres Distributed* .

You can configure PEM to display status information about one or more EDB Postgres Distributed database nodes using dashboards in PEM version 8.1.0 and EDB Postgres Distributed version 3.7.9 and later.

To configure PEM to monitor EDB Postgres Distributed database nodes, use the PEM web client to create a server definition. Use the tabs on the *Manually registering a database server* dialog box to specify general connection properties for the EDB Postgres Distributed database node with the following exceptions:

- Specify the EDB Postgres Distributed-enabled database name in the **Database** field of the **PEM Agent** tab.
- Specify the user having the pgd_monitor or pgd_superuser role in the **username** field of the **PEM Agent** tab.

After saving the server definition, the EDB Postgres Distributed database node is included in the list of servers under the PEM server directory in the PEM client object browser tree. You can monitor the nodes from EDB Postgres Distributed (PGD) dashboards.

To include monitoring information on the EDB Postgres Distributed (PGD) dashboards, you must enable the relative probes for each EDB Postgres Distributed group. See the complete list of EDB Postgres Distributed *Probes* .

To enable a probe, right-click the node name, and select **Management > Manage Probes**.

To monitor the EDB Postgres Distributed database node, right-click the name of the node in the object browser tree. From the **Dashboards** menu, select the PGD Admin, PGD Group Monitoring, or PGD Node Monitoring dashboard.

CHAPTER 29

Monitoring Failover Manager

If you're using EDB Failover Manager to monitor your replication scenario, you must manually install and configure Failover Manager. For detailed information about installing Failover Manager, see the EDB website .

To monitor the status of a Failover Manager cluster on the Streaming Replication dashboard, you must provide the following information on the **Advanced** tab of the **Server Properties** dialog box for each node of the cluster:

- Use the **EFM Cluster Name** field to specify the name of the Failover Manager cluster. The cluster name is the prefix of the name of the cluster properties file. For example, if your cluster properties file is named `efm.properties` , your cluster name is efm.
- Use the **EFM Installation Path** field to specify the location of the Failover Manager binary file. By default, the Failover Manager binary file is installed in `/usr/edb/efm-<X>/bin` , where `<X>` is the EFM version.

Note

To monitor Failover Manager, the PEM agent executes the *efm cluster-status-json <cluster_name>* command as the user efm. This command fails if the cluster properties file isn't in the efm user's home directory. In this case, create a symlink to the cluster properties files in the home directory of efm, and ensure that efm has permission to read and execute the cluster properties file.

After registering your servers, the Streaming Replication Analysis dashboard displays status information about your EFM cluster near the bottom of the dashboard.

The **Failover Manager Cluster Status** section of the Streaming Replication Analysis dashboard displays information about the monitored cluster.

The Failover Manager Cluster Information table provides information about the Failover Manager cluster:

- The Properties column displays the name of the cluster property.
- The Values column displays the current value of the property.

The Failover Manager Node Status table displays information about each node of the Failover Manager cluster:

- The Agent Type column displays the type of agent that resides on the node. The possible values are Primary, Replica, Witness, Idle, and Promoting.

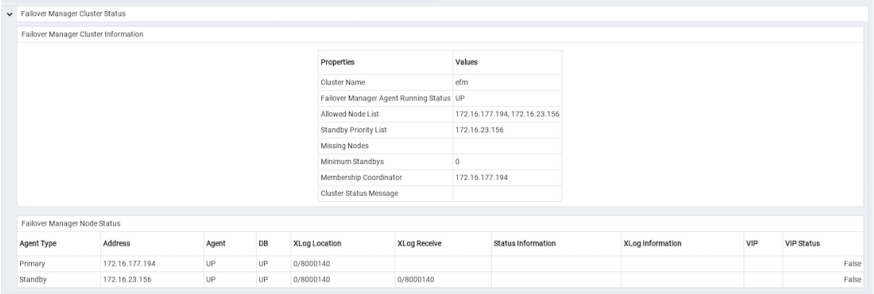


Fig. 1: The Failover Manager cluster status report

- The Address column displays the IP address of the node.
- The Agent column displays the status of the agent that resides on the node.
- The DB column displays the status of the database that resides on the node.
- The XLog Location column displays the transaction log location of the database.
- The Status Information column displays any error-related information about the node.
- The XLog Information column displays any error-related information about the transaction log.
- The VIP column displays the VIP address that's associated with the node.
- The VIP Status column displays True if the VIP is active for the node, False if not.

29.1 Replacing a primary node

You can use the PEM client to replace the primary node of a Failover Manager cluster with a replica node. To start the failover process, select **Tools > Server > Replace Cluster Primary**. You are prompted to confirm that you want to replace the current primary node.

Select **Yes** to remove the current primary node from the Failover Manager cluster and promote a replica node to the role of read/write primary node in a Failover Manager cluster. The node with the highest promotion priority (defined in Failover Manager) becomes the new primary node. PEM reports the job status.

When the job completes and the Streaming Replication Analysis dashboard refreshes, you can review the Failover Manager Node Status table to confirm that a replica node was promoted to the role of primary in the Failover Manager cluster.

29.2 Switchover EFM cluster

You can use the PEM client to replace the primary node of a Failover Manager cluster with a replica node. To start the switchover process, select **Tools > Switchover EFM Cluster**. You are prompted to confirm that you want to switch over EFM cluster. Select **Yes** to:

- Start the Failover Manager switchover.
- Promote a replica node to the role of read/write primary node.
- Reconfigure the primary database as a new replica in a Failover Manager cluster.

The node with the highest promotion priority (defined in Failover Manager) becomes the new primary node. PEM reports the job status. When the job completes and the Streaming Replication Analysis dashboard refreshes, you can review the Failover Manager Node Status table to confirm that a switchover occurred.

CHAPTER 30

Monitoring Replication Server

Before configuring PEM to retrieve statistics from an EDB Postgres Advanced Server or PostgreSQL database that's part of an xDB replication scenario, you must manually install and configure xDB replication. For more information about xDB replication solutions and documentation, see Multi-master replication operation .

The PEM xDB Replication probe monitors lag data for clusters that use xDB multi-primary or single-primary replication that have an EDB Postgres Advanced Server or PostgreSQL publication database. If you configured replication between other proprietary database hosts (that is, Oracle or SQL Server) and EDB Postgres Advanced Server or PostgreSQL, the probe can't return lag information.

By default, the xDB Replication probe is disabled. To enable the xDB Replication probe:

1. Right-click the name of the server and select **Connect** from the context menu.
2. If prompted, provide authentication information.
3. After connecting, expand the server node of the tree and select the name of the replicated database.
4. Select **Management > Manage Probes**.

Use the **Manage Probes** tab to configure the xDB Replication probe:

- Set **Default** to **No** to modify the minutes and seconds between probe executions.
- Use the **Enabled?** slider to enable the xDB Replication probe.
- Set **Default** slider in the **Data Retention** field to **No** to modify the number of days for PEM to store the information retrieved by the probe.

After enabling the probe, you can use the metrics returned to create custom charts and dashboards in the PEM client.

CHAPTER 31

Monitoring event history

You can monitor the executed event details in the `pem.event_history` table. It provides the username, execution time of the event, type of the event occurred, and the details regarding the event. The fields in the table are:

!!!Note Currently, `pem.event_history` table records only the alert blackout history.

Field name	Description
recorded_time	Records the execution time of the event. Displays the date and time along with the fractional seconds.
user_name	The username of the user who executed or scheduled the event.
component	The event name.
operation	The action taken for the event. For example, if the user has enabled an alert blackout, it displays *enable_alert_blackout*.
message	The description of the action taken for the event. For example, if the user has enabled an alert blackout for the server, it displays *Enabled the alert blackout for the server*.
details	The JSON output of the event that occurred.

CHAPTER 32

Tuning performance

The following features help in tuning performance:

- *Performance Diagnostic*

32.1 Performance Diagnostic

The Performance Diagnostic dashboard analyzes the database performance for Postgres instances by monitoring the wait events. To display the diagnostic graphs, PEM uses the data collected by the EDB wait states module. For more information on EDB wait states, see EDB wait states .

To analyze the wait states data on multiple levels, narrow down the data you select. The data you select at the higher level of the graph populates the lower level.

> **Note**
>
> The Performance Diagnostic dashboard is available for all the Postgres variants and versions for which the EDB wait states module is available.

32.1.1 Prerequisites

- Install the EDB wait states package:

```
-  For PostgreSQL, see  `EDB repositories <https://repos.enterprisedb.com/>`_  .

-  For EDB Postgres Advanced Server, see  :ref:`EDB wait states </pg_extensions/
↪wait_states/installing/>`  .
```

- After you install the EDB wait states module of EDB Postgres Advanced Server: 1. Configure the list of libraries in the `postgresql.conf` file:

```
shared_preload_libraries = $libdir/edb_wait_states
```

2. Restart the database server. 3. Create the following extension in the maintenance database:

```
CREATE EXTENSION edb_wait_states;
```

- If you want to access the Performance Diagnostic dashboard as a user without superuser privileges, specific *Login roles* must be granted to the user based on the server to which the user is connected or logged into:
- Grant `pem_comp_performance_diagnostic` role to the user (`test1` in this example), who is logged into the PEM server.

```
GRANT pem_comp_performance_diagnostic TO test1;
```

```
- Grant `pg_monitor` role and `EXECUTE` privilege on all functions of `edb_wait_
↪states` to the database user (`test2` in this example). The `test2` user is␣
↪connected to the database server with EDB wait states.
```

```
# Grant `pg_monitor` role
GRANT pg_monitor TO test2;

# Grant `Execute` privilege on all functions
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;
```

If the prerequisites aren't met, an error appears when you access the Performance Diagnostic dashboard.

32.1.2 Using the Performance Diagnostic dashboard

To open the Performance Diagnostic dashboard, on the PEM client select **Tools > Server > Performance Diagnostic**.

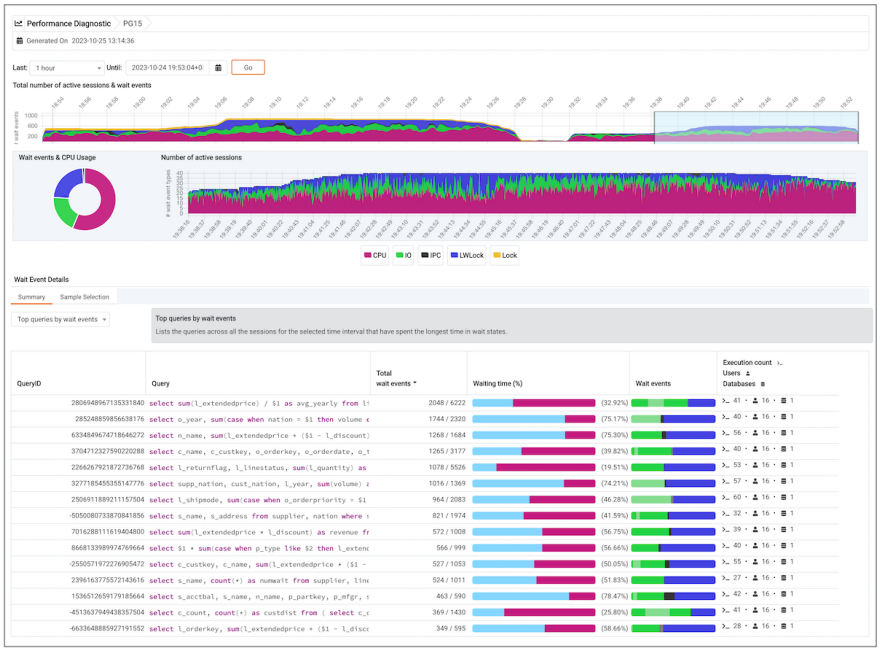


Fig. 1: Performance Diagnostic dashboard

By default, the top Performance Diagnostic graph pulls the data of the last hour, starting from the current date and time. This graph shows the time series containing the number of active sessions. Each point of this time series represents the sum of wait events and CPU usage over the previous 15 seconds. These sessions might be waiting for a wait event or using the CPU at a particular point in time. This time series is generated based on the wait event samples collected by the EDB Wait States extension.

To open the Performance Diagnostic dashboard in a new browser tab, select **Preferences > Open in New Browser Tab?**.

Performance Diagnostic dashboard

- The first graph displays the number of active sessions and wait event types for the selected duration. You can change the duration in the first graph to analyze the data for a specific time period.

By default, the duration selection in the first graph is 15 minutes. The duration can be extended up to one hour. To see the duration on a graph, select a duration from the **Last** list. To display the data for a specified date and time, select a date and time from the **Until** list.

- The next section plots the following graphs based on the selected duration in the first graph:
- Donut graph _ Shows total wait event types and CPU usage according to the duration selection in the first graph. This graph can provide a better understanding of how much time those sessions spent waiting for an event.
  - Line graph _ Plots a time series with each point representing the active sessions for each sample time.

To differentiate each wait event type and the CPU usage more clearly, the graph for each wait event type displays in a different color.

Select a time on the line graph to analyze the wait events. To show or hide a wait event type in all the graphs, select the graph legends. The analysis is simpler when you can see only the wait event types you want to analyze.

- The third section displays the wait event details in the Performance Diagnostic dashboard based on your selected duration in the second graph. It displays the wait event details on two tabs:
- The **Summary** tab displays the list of top 15 SQL queries with query IDs, total CPU events, percentage of CPU usage, wait events, and execution count for the selected time interval. These SQL queries are grouped into four categories:
  - **Top queries by wait events** _ Lists the queries across all the sessions for the selected sample time that have spent the longest time in wait states. It displays the query ID, query, total wait events, waiting time in percentage, wait events, and the execution count along with number of users that executed that query in the number of databases.
  - **Top queries by CPU usage** _ Lists the queries across all the sessions for the selected sample time that used the most CPU time. It displays the query ID, query, total CPU events, CPU usage percentage, wait events, and execution count along with the number of users that executed that query in the number of databases. The data in the table is in decreasing order of the number of CPU events.
  - **Top users by wait events** _ Lists the users across all the sessions for the selected sample time whose queries have spent the longest time in wait states. It displays the usernames, database name, total wait events, waiting time percentage, wait events, and the number of queries that have spent the longest time in wait states.
  - **Top users by CPU usage** _ Lists the users across all the sessions for the selected sample time whose queries used the most CPU time. It displays the username, database name, total CPU events, CPU usage percentage, wait events, and the number of queries that used the most CPU time.
- The **Sample Selection** tab displays the details of the wait events for the selected sample time in the second graph. The wait events are grouped by:

- **Queries** _ Lists the queries along with query IDs and number of sessions that executed the query in the selected sample time. Select the eye in any row to display a window with details about that query. For more information, see *Query information* .
  - **Users** _ Lists the details of the wait events grouped by users that executed the queries along with the number of events and execution count of the query.
  - **Waits** _ Lists the number of wait events belonging to each wait event type for the selected sample time. You can filter the data displayed in the rows under both the tabs. You can also sort the data alphabetically by selecting the column headers.

Query information

To open the query information window, select the eye in any row of the **Queries** grouping under the **Sample Selection** tab. This window displays in the **Query information** section a query ID and its corresponding session IDs in a list at that selected sample time. You can select the **Session ID** list for the selected query whose data you want to analyze. The details corresponding to the selected session ID and query ID appear. The **Query information** table also displays the SQL query.

The **Wait event types** section displays the total number of wait event types for the selected session ID and query ID. It shows two types of graphs:

- Donut graph _ Shows the proportions of categorical data.
- Timeline bar graph _ Visualizes trends in counts of wait event types over time.

To differentiate, each wait event type is represented by a different color in the bar graph.

The **Wait events** section has a table displaying all the wait events that occurred during the query execution. It displays data in decreasing order by number of wait events. The second table displays the wait event with sample times that occurred over the period of the whole query execution. It allows you to analyze the wait events during the query execution over the period of time. Also, it shows the actual samples collected by the EDB Wait States extension for that query ID and session ID.

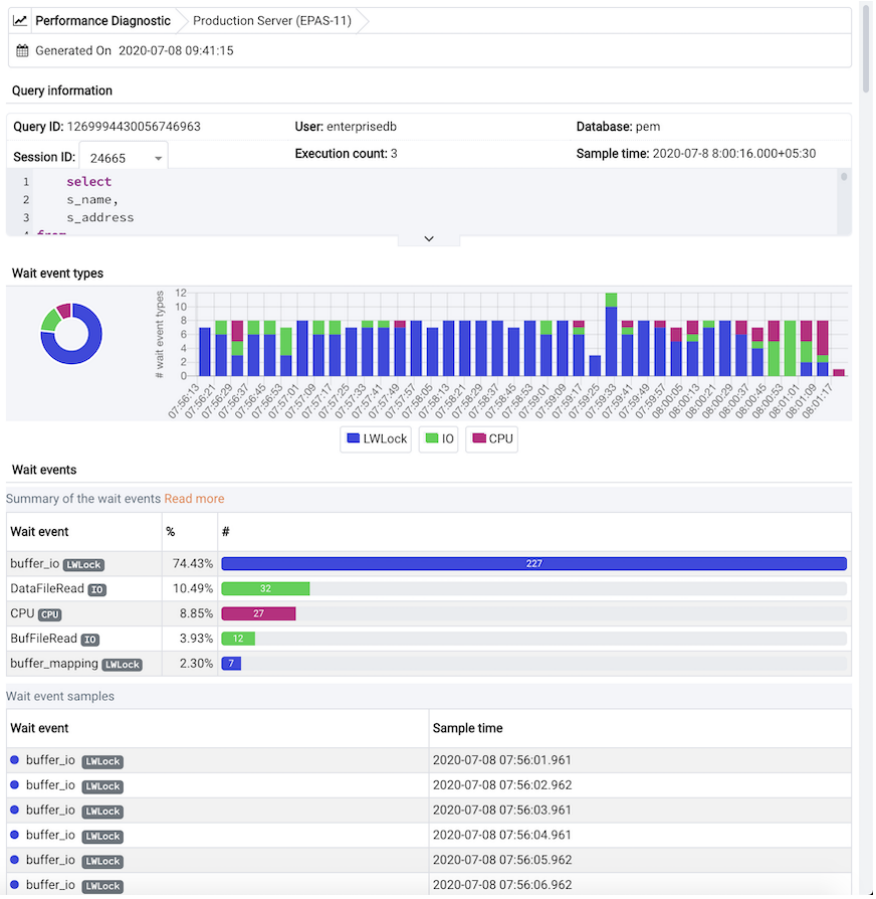


Fig. 2: Performance Diagnostic - Query dashboard

CHAPTER 33

Profiling workloads

The SQL Profiler extension works with PEM to allow you to profile a server's workload in fine detail. Use SQL Profiler to collect query plans, buffer usage, execution times etc.

- See *Installing the SQL Profiler extension*
- See *Upgrading SQL Profiler*
- See *Using SQL Profiler*

33.1 Installing the SQL Profiler extension

You must install and enable the SQL Profiler extension on each server on which you want to use it. For example, if you have a host running PostgreSQL 14 and PostgreSQL 15, you must enable the extension on each server.

SQL Profiler is supported on the same platforms as the Postgres distribution you're using. See:

- EDB Postgres Advanced Server Product Compatibility
- PostgreSQL Product Compatibility
- EDB Postgres Extended Server Product Compatibility

33.1.1 Installing the package

Prerequisites

Before you begin the installation process:

- Install Postgres. See: - Installing EDB Postgres Advanced Server
- Installing PostgreSQL
- Installing EDB Postgres Extended Server
- Set up the repository.

Setting up the repository is a one-time task. If you have already set up your repository, you don't need to perform this step.

To set up the repository, go to EDB repositories and follow the instructions provided there.

Install the package

The syntax for the install command is:

```
sudo <package-manager> -y install edb-<postgres><postgres_version>-server-sqlprofiler
```

Where:

- `<package-manager>` is the package manager used with your operating system:

Package manager	Operating system
dnf	RHEL 8 and derivatives
zypper	SLES
apt-get	Debian and derivatives

- `<postgres>` is the distribution of Postgres you're using:

Postgres distribution	Value
PostgreSQL	pg
EDB Postgres Advanced Server	as
EDB Postgres Extended	postgresextended

- `<postgres_version>` is the version of Postgres you're using.

For example, to install the latest version of SQL Profiler for EDB Postgres Advanced Server 15 on a RHEL 8 platform:

```
sudo dnf -y install edb-as15-server-sqlprofiler
```

33.1.2 Enabling the extension

To enable the extension:

1. Edit the `postgresql.conf` file on the server you want to profile, modifying the `shared_preload_libraries` parameter:

```
shared_preload_libraries = $libdir/sql-profiler
```

1. Restart the Postgres server.
2. Create the SQL Profiler extension in your maintenance database with the following command:

```
# Connect to your maintenance database using psql
psql -d postgres -U enterprisedb
# Create the extension
CREATE EXTENSION sql_profiler;
```

> **Note**
>
> If you are connected to the PEM server with the PEM client before configuring SQL Profiler, you must disconnect and reconnect with the server to enable SQL Profiler functionality.

After enabling the extension, SQL Profiler is ready to use with all databases that reside on the server.

33.2 Using SQL Profiler

The SQL Profiler extension allows a user to locate and optimize inefficient SQL code. Microsoft's SQL Server Profiler is very similar to PEM's SQL Profiler in operation and capabilities.

SQL Profiler works with PEM to allow you to profile a server's workload. You can install and enable the SQL Profiler extension on servers with or without a PEM agent. However, you can run traces only in ad hoc mode on unmanaged servers and you can schedule them only on managed servers.

SQL Profiler captures and displays a specific SQL workload for analysis in a SQL trace. You can start and review captured SQL traces immediately or save captured traces for review later. You can use SQL Profiler to create and store up to 15 named traces.

33.2.1 Permissions for SQL profiler

To access the SQL profiler tool on PEM, there are two prerequisites:

1. The user logged in to the PEM GUI must either be a superuser or a member of group `pem_comp_sqlprofiler` within the PEM server database. For more information, see

pem groups .

2. The user configured for the database server in the server tree (`Username`), must be a superuser on the database server that the trace is being run on (monitored server).

In SQL Profiler version 4.2.0 and later, the `sql_profiler_admin` role allows a non-superuser to execute SQL Profiler functions. To assign this role to a user, run the following command:

```
grant sql_profiler_admin to username
```

33.2.2 Creating a trace

You can use the Create Trace dialog box to define a SQL trace for any database on which SQL Profiler was installed and configured. To open the dialog box, select the database in the PEM client tree and select **Tools > Server > SQL Profiler > Create trace**.

Use the **Trace options** tab to specify details about the new trace:

- Provide a name for the trace in the **Name** field.
- Use the **User filter** field to specify the roles whose queries to include in the trace. Select **Select All** to include queries from all roles.
- Use the **Database filter** field to specify the databases to trace. Select **Select All** to include queries against all databases.
- Specify a trace size in the **Maximum Trace File Size** field. SQL Profiler terminates the trace when it reaches approximately the size specified.
- Select **Yes** in the **Run Now** field to start the trace when you select **Create**. Select **No** to enable fields on the **Schedule** tab.

Use the **Schedule** tab to specify scheduling details for the new trace:

- Use the **Start time** field to specify the starting time for the trace.
- Use the **End time** field to specify the ending time for the trace.
- Select **Yes** in the **Repeat?** field to repeat the trace every day at the times specified. Select **No** to enable fields on the **Periodic job options** tab.

Use the **Periodic job options** tab to specify scheduing details about a recurring trace. Use the **Days** section to specify the days when the job executes:

- Use the **Week days** field to select the days of the week for the trace to execute.
- Use the **Month days** field to select the days of the month for the trace to execute.
- Use the **Months** field to select the months when the trace executes.

Use the **Times** section to specify a schedule for the trace execution. Use the **Hours** and **Minutes** fields to specify the time when the trace executes.

After you complete the Create Trace dialog box, select **Create** to start the trace or to schedule the trace.

If you execute the trace immediately, the trace results appear in the PEM client.

33.2.3 Opening an existing trace

To view a previous trace, select the profiled database in the PEM client tree and select **Management > SQL Profiler > Open trace**. The Open Trace dialog box opens.

Select an entry in the trace list and select **Open** to open the selected trace in the **SQL Profiler** tab.

33.2.4 Filtering a trace

A filter is a named set of one or more rules, each of which can hide events from the trace view. When you apply a filter to a trace, the hidden events aren't removed from the trace but are excluded from the display.

Select **Filter** to open the Trace Filter dialog box and create a rule or set of rules that define a filter. Each rule screens the events in the current trace based on the identity of the role that invoked the event or the query type invoked during the event.

To open an existing filter, select **Open**. To define a new filter, select **Add (+)** to add a row to the table displayed on the **General** tab and provide rule details:

- Use the **Type** list to specify the trace field that the filter rule applies to.
- Use the **Condition** list to specify the type of operator for SQL Profiler to apply to the value when it filters the trace:
- Select **Matches** to filter events that contain the specified value.
- Select **Does not match** to filter events that don't contain the specified value.
- Select **Is equal to** to filter events that contain an exact match to the string specified in the **Value** field.
- Select **Is not equal to** to filter events that don't contain an exact match to the string specified in the **Value** field.
- Select **Starts with** to filter events that begin with the string specified in the **Value** field.
- Select **Does not start with** to filter events that don't begin with the string specified in the **Value** field.
- Select **Less than** to filter events that have a numeric value less than the number specified in the **Value** field.
- Select **Greater than** to filter events that have a numeric value greater than the number specified in the **Value** field.

- Select **Less than or equal to** to filter events that have a numeric value less than or equal to the number specified in the **Value** field.
- Select **Greater than or equal to** to filter events that have a numeric value greater than or equal to the number specified in the **Value** field.
- Use the **Value** field to specify the string, number, or regular expression to search for.

After you finish defining a rule, select **Add (+)** to add another rule to the filter. To delete a rule from a filter, select the rule and select **Delete**.

Select **Save** to save the filter definition to a file without applying the filter. To apply the filter, select **OK**.

33.2.5 Deleting a trace

To delete a trace:

1. Select the profiled database in the PEM client tree.
2. Select **Management > SQL Profiler > Delete trace(s)**.
3. In the Delete Traces dialog box, select the icon to the left of a trace name to mark one or more traces for deletion.
4. Select **Delete**. The PEM client acknowledges that the selected traces were deleted.

33.2.6 Viewing scheduled traces

To view a list of scheduled traces, select the profiled database in the PEM client tree. Select **Management > SQL Profiler > Scheduled traces**.

The Scheduled Traces dialog box displays a list of the traces that are awaiting execution. Select **Edit** to the left of a trace name to see detailed information about the trace:

- The **Status** field lists the status of the current trace.
- The **Enabled?** switch displays Yes if the trace is enabled, No if it is disabled.
- The **Name** field displays the name of the trace.
- The **Agent** field displays the name of the agent responsible for executing the trace.
- The **Last run** field displays the date and time of the last execution of the trace.
- The **Next run** field displays the date and time of the next scheduled trace.
- The **Created** field displays the date and time that the trace was defined.

CHAPTER 34

Reports

Manage and review diagnostic data for your monitored servers. Postgres Enterprise Manager (PEM) provides two distinct reporting frameworks to suit your diagnostic needs.

34.1 Custom report templates

The **Custom Reports** framework allows you to design and execute bespoke diagnostic templates. By combining SQL queries and host-level scripts, you can extract specific insights tailored to your unique infrastructure requirements.

- **User-defined logic** _ Build templates from the ground up using SQL or shell/batch scripts.
- **Hierarchical structure** _ Organize data into logical sections and items for clear reporting.
- **Portability** _ Export and import templates via JSON for easy migration across environments.

34.2 Viewing system reports

System Reports are the pre-defined, built-in diagnostic tools provided by PEM. These reports offer out-of-the-box analysis of database performance, security audits, and configuration states.

- **Standardized metrics** _ Access industry-standard reports for database health and activity.
- **No configuration required** _ Execute immediate diagnostics without the need for custom script writing.
- **Deep integration** _ Fully integrated with PEM's alerting and monitoring history.

> **Note**
>
> To run a report, simply right-click a **Target Server** in the PEM Browser tree and navigate to the relevant report category.

34.2.1 Custom report templates

The **custom report templates** feature in PEM 10.4 introduces a robust framework that enables users to design, manage, and execute tailored diagnostic templates across any server actively **bound to a PEM Agent**. By consolidating SQL queries and host-level batch or shell scripts into organized logical sections, this feature allows users to extract highly specific insights with results rendered in versatile **Table** (tabular grid) or **Text** (scalar value) formats.

Prerequisites

To execute scripts or SQL queries via custom report templates:

- The target server must be bound to a PEM agent (version 10.4 or later).
- For shell/batch scripts, the agent must have the necessary permissions to execute commands on the host operating system.

Managing report templates

To access the management interface, navigate to **Management** > **Reports** > **Manage Report Templates**.

Creating a custom report template:

The top-level container defined by a unique title and functional description.

- Select **Create New Report**.
- **Title:** Enter a unique name for the report (required).
- **Description:** Provide an explanation of the report's purpose (optional).

Defining the report structure

Custom report templates utilize a hierarchical architecture to ensure data is structured and logically categorized:

- **Report**: The top-level container; contains one or more **Sections**.
- **Section**: A logical grouping within a report; contains one or more **Items**.
- **Item**: The individual unit of execution.

Item definitions: Each item within a section is defined by the following attributes.

Name	Description
Title	The name of the specific item.
Description	An optional field providing functional context or an explanation.
Kind	Specifies the execution engine: **SQL Query** or **Batch/Shell Script** .
Format	Defines the presentation layer: **Table** (tabular output) or **Text** (single value).
Code	The specific SQL statement or script logic to be executed by the agent.

Executing a report

- Select the **Target Server** in the PEM browser (ensure it is bound to a 10.4 agent).
- Right-click the server and select **Run Report**.
- Select the desired **Report Template** and the target **Database** from the dropdown menus.
- Select **Run**. This triggers an agent-level job to execute the report.

> **Note**
>
> Custom reports cannot be executed on the **PEM database** itself.

Viewing execution history

To monitor the status of triggered reports and audit historical data, navigate to **Management** > **Reports** > **Report Execution History**.

The execution history provides a centralized audit trail of all custom report templates activities. To effectively manage and locate specific records within a high-volume environment, you can apply the following filters:

- **Server Name:** Filter by the specific target server where the report was executed.
- **Agent Name:** Filter by the unique PEM Agent responsible for processing the execution.
- **Status:** Filter by execution states (e.g., **Success**, **Partially Successful**, **Failed**, **In Progress**, or **Unknown**).
- **Time Range:** Define a specific temporal window to isolate recent or historical data.

Viewing report results

To access the detailed output of a report, select any execution record within the **Report Execution History** dashboard.

Result organization: Results are structured according to the original template hierarchy defined during the creation phase. Data is partitioned into **Sections**, with each section containing its respective **Items**.

Data presentation layers: Depending on the **Format** attribute assigned to each item within the template, results are rendered in one of two ways:

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.

Handling errors: If an item within a report fails to execute, the system captures and displays the specific error message returned by the SQL engine or the shell environment directly within the item's result block. This ensures that even in a **Partially Successful** report, you can identify and troubleshoot individual point failures.

Exporting results: For external reporting or further data processing, you can download the executed report in the following formats:

- **HTML:** A formatted document for easy sharing and browser-based review.
- **JSON:** A structured data file optimized for integration with automated workflows or third-party tools.

Importing and exporting reports

PEM allows for the seamless migration and backup of report templates via standardized JSON files.

Exporting report templates To backup or share your report configurations:

- Navigate to **Management** > **Reports** > **Manage Report Templates**.
- Select one or more templates from the list.
- Select **Export** to download the selected configurations as a JSON file.

Importing report templates To bulk-add templates to a new or existing PEM instance:

- Open the **Manage Report Templates** interface.
- Select the **Import** button.
- Upload a valid JSON file containing report template definitions.

> **Note**
>
> If a template with the same title already exists, you will be prompted to either skip or overwrite the entry.

Understanding status definitions

The **Report execution history** tab tracks the lifecycle of each report job. The following table defines the possible states for a custom report templates:

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.

Limiting SQL row counts

To optimize performance and prevent excessive memory consumption on the PEM server, custom report templates enforce a limit on the number of rows returned by SQL queries.

This limit is managed globally via the **Server Configuration** dashboard, where administrators can modify the value to accommodate larger data sets.

Parameter	Default Value	Description
report_sql_row_limit	500	Caps the maximum number of rows returned by any single SQL query item within a report.

> **Note**
>
> Increasing this limit may significantly impact report generation speed and the overall responsiveness of the **Report results** view when handling large volumes of data.

System defined report templates

While PEM allows for fully customized reporting, it also includes pre-configured system report templates designed for deep database analysis. These reports are built into the platform to provide immediate value without requiring manual template configuration.

EDB Postgres workload report (PWR)

The EDB Postgres workload report (PWR) provides a comprehensive overview of a server's performance and resource utilization over a specific snapshot duration. It is an essential tool for identifying bottlenecks, high-load SQL statements, and wait events.

> **Note**
>
> To execute this report template, ensure the *edb_wait_states* extension is installed and enabled on the target database.

Server information

Shows information about the server version:

- **Server version** _ The version number of the Postgres server.
- **Architecture** _ The hardware architecture for which the server is built.
- **System identifier** _ A unique identifier for the database cluster.
- **Redwood mode** _ Indicates whether the cluster was created in Redwood (Oracle compatibility) mode.
- **Current user** _ The user who generated the report.
- **Actual start/end snap timestamp** _ The precise timestamps marking the beginning and end of the data collection.
- **Snapshot duration** _ The total time elapsed during the snapshot.

Shows information about the report timestamps:

- **Session counts** _ The number of connected sessions at both the start and end of the snapshot.
- **Database time** _ The total non-idle time (in seconds) that all sessions spent either working or waiting in the database.
- **Uptime** _ Total time spent since the last server restart.
- **Role** _ Indicates whether the server is a Primary or Standby instance.

Load profile

Shows the average load profile per transaction:

- **Blocks fetched** _ Average number of blocks accessed per transaction.
- **WAL records/bytes** _ Average number of Write-Ahead Log records and bytes generated per transaction.
- **Transactions** _ The total number of transactions processed during the snapshot.
- **Abort rate** _ The percentage of transactions that resulted in an abort.

Top wait events

Displays total wait time spent on top events (in seconds):

- **Wait event** _ The name of the specific wait event. (**CPU** indicates time spent actively working).
- **Wait class** _ The category or type of the wait event.
- **Wait time** _ Total time in seconds spent on this event.
- **Pct database time** _ The percentage of total `Dbtime` spent waiting on this event type.

Top SQL statements

Displays the top 10 SQL statements sorted by `dbtime` :

- **Database time** _ The cumulative database time (in seconds) consumed during the execution of the SQL statement.
- **Wait time** _ The total duration the SQL statement spent waiting for system resources or background events.
- **CPU time** _ The total amount of CPU processing time utilized by the SQL statement.
- **Top wait event** _ The specific wait event identifier where the statement experienced its longest delay.
- **Query** _ The actual SQL text of the executed query.

Transaction stats

Shows transaction stats during the snapshot duration:

- **Xact commit** _ The total number of successfully completed and saved transactions.
- **Xact rollback** _ The total number of reversed or aborted transactions.

WAL stats

Shows stats about WAL record changes during snapshot duration:

- **Wal records** _ The total number of individual WAL records generated during the snapshot.
- **Wal fpi** _ The number of Full Page Images (FPI) created; these occur after the first change to a page following a checkpoint and can significantly increase WAL volume.
- **Wal bytes** _ The total size of all generated WAL data, measured in bytes.

Shared buffers stats

Shows shared buffers stats during the snapshot duration:

- **Blocks read** _ The total number of disk blocks read.
- **Blocks hit** _ The total number of blocks hit.

Tuple stats

Shows stats about the tuples during the snapshot duration:

- **Tuples returned** _ The number of rows returned by sequential scans.
- **Tuples fetched** _ The number of rows fetched specifically via index scans.
- **Tuples inserted** _ The total number of new rows added to the database.
- **Tuples updated** _ The total number of existing rows modified.
- **Tuples deleted** _ The total number of rows removed from the database.

Temporary files stats

Shows stats about the temporary file creation:

- **Temp files** _ The number of temporary files created.
- **Temp bytes** _ The total volume of data written to temporary files.

User session information

Session information connected/running during snapshot period:

- **Session ID** _ The unique ID assigned by `edb_wait_states` to a specific Postgres session.
- **Database name** _ The name of the database to which the session is currently connected.
- **Username** _ The identity of the database user logged into the session.
- **Application name** _ The name of the application or client tool connected to this session (e.g., `psql` , `pgAdmin` , or a custom app service).
- **Client hostname** _ The network host name of the connected client, providing the origin of the connection (where available).
- **Session start timestamp** _ The precise date and time when the session was initially established.

System information

Shows CPU and memory information of the server machine:

- **Host name** _ The network name of the machine where Postgres is running.
- **CPU information** _ Details including the number of processors, vendor, and clock speed.
- **Memory information** _ Total memory information for the host.

Postgres database settings

Shows the settings of the Postgres database:

- **Category** _ The logical functional group of the parameters.
- **Parameter** _ The name and current value of settings within a category. A highlighted entries denotes parameters with custom values.
- **Value** _ The current operational setting for the specified parameter.
- **Is custom value** _ A flag indicating if the parameter has been modified from its default value.

34.2.2 Viewing system reports

You can generate the System Configuration report and Core Usage report for all locally and remotely managed servers. To generate this report, select **Management > Reports**.

Reports has the following options:

- Alert History Report (JSON)
- System Configuration Report (JSON)
- System Configuration Report (HTML)
- Core Usage Report (JSON)
- Core Usage Report (HTML)

Only superusers or the users with the pem_admin role can download the System Configuration or Core Usage reports.

Information in these reports shows the latest probe run time.

Alert History report

The Alert History report provides detailed information about the alerts history at the agent or server level in JSON format.

Select **Management > Reports > Alert History Report**. From the dialog box, select your options:

- **Agent/Server Name** _ Select the agents and servers from the list. The **Overall System Report** option is also available to generate a report for all the registered and active agents and servers.
- **Timeframe** _ Select the required timeframe from the list.
- **Alert Types** _ Select the alert types you want to generate the alert history report for.

System Configuration report

The System Configuration report provides detailed information about the PEM Agents group, PEM Server Directory group, and custom groups listed under the browser tree. These groups can contain PEM server, PEM agent, and database servers. You can download this report in HTML and JSON formats.

The Postgres Enterprise Manager Summary provides details about:

- The Postgres Enterprise Manager backend database server version
- Application version
- User name accessing the application
- Python version
- Flask version
- Platform specific information

The summary provides information about the number of agents and servers. The Group: PEM Agents panel provides details about the PEM agent, CPU cores, disk utilization, and memory information.

The Group: PEM Server Directory panel provides details about:

- Database server version
- Host
- Port
- Database name
- Database size
- Tablespace size

The group server name depends on the group name to which the server is added.

Core Usage report

The Core Usage report provides detailed information about the number of cores specific to:

- Server type
- Database version
- Platform and group name

The report also gives detailed information about locally managed servers:

- Type
- Host

- Port
- Platform
- Cores
- RAM

CHAPTER 35

PEM command line interface

The PEM CLI allows you to export and import the alert templates or probes from one PEM server to another.

You can install the PEM CLI using the package `edb-pem-cli-8.5.0-1-<platform_name>` separately from the `edb.repo` repository. It creates the binary file `/usr/edb/pem/cli/bin/pem` and the license file `/usr/edb/pem/edb-pem-cli-license.txt` .

After installing the PEM CLI package, use the `./pem -h` command to see all the available options.

```
$ ./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
```

(continues on next page)

(continued from previous page)

```
  -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.

                              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.
```

You can see all the available options with the `import-alert-templates` subcommand:

```
$ ./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
```

You can see all the available options with the `list-alert-templates` subcommand:

```
$ ./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
```

You can see all the available options with the `export-alert-templates` subcommand:

```
$ ./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
```

You can see all the available options with the `import-probes` subcommand:

```
$ ./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
```

You can see all the available options with the `export-probes` subcommand:

```
$ ./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
```

You can see all the available options with the `list-probes` subcommand:

```
$ ./pem list-probes -h
__OUTPUT__
List the probes in the PEM server.
Usage: ./pem list-probes [OPTIONS]

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
```

35.1 Example

To list all the probes (system as well as custom probes) use the `-c all` option:

```
$ PEM_PASSWORD_FILE=/tmp/pass PEM_API_USER=postgres PEM_SERVER_URL=https://172.17.0.4/
↪pem ./pem -i list-probes -c all
```

Where:

- `PEM_PASSWORD_FILE=/tmp/pass` is the password file with the password of the postgres user.
- `PEM_API_USER=postgres` is the username through which you're running the PEM CLI.
- `PEM_SERVER_URL=https://172.17.0.4/pem` is the URL of the PEM server containing the probes.
- `-i list-probes` is the `insecure` option with the `list-probes` subcommand to allow insecure connection and to list the probes.
- `-c all` is the `-c` option with the `all` value to list all the system as well as custom probes.

CHAPTER 36

EDB Postgres Enterprise Manager REST APIs v1.0.0

EDB Postgres Enterprise Manager uses a token-based authentication mechanism to allow access to exposed REST APIs.

PEM uses tokens for authentication purposes. To generate a token, please refer the section Token. After receiving a token string, you should include the token string in the 'X-Auth-Token' header when calling any resources except https://PEM-SERVER-IP/api/token/.

NOTE:

- **EDB Postgres Enterprise Manager** *is referred as* **PEM** *in this documentation.*
- **PEM** *uses self-certified SSL certificates, so you may encounter errors such as 'SSL certificate problem: self signed certificate'. You may need to use an 'insecure' connection to connect to the server.

Rest api versions:

- v16 (Current)

Base URLs:

- https://PEM-SERVER-IP/pem/api

Email: EnterpriseDB Web: EnterpriseDB License: Limited Use Software License Agreement

36.1 Token

A string generated and used by PEM REST APIs for authentication purposes

36.1.1 TokenGeneration

Code samples

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/token/ \
```

(continues on next page)

(continued from previous page)

```
  -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/`

- Generate a token for authentication.*

This URL creates a token to be used by the other REST APIs exposed by PEM. You must provide a valid username and password, which will be validated during the process.

The token will be valid for 20 mins.

Body parameter

```
{
  "username": "string",
  "password": "string"
}
```

36.1.2 Parameters

Name	In	Type	Required	Description
body	body	object	true	Creates the token using the /token/ URL first.
» username	body	string	true	Validates username, which has pem_user role.
» password	body	string	true	Password for the username

Example responses


```
{
  "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"
  ]
}
```


36.1.3 Responses

Status	Meaning	Description	Schema
201	Created	Token was created and can be found as 'X-Subject-Token' in the response header.	Inline
401	Unauthorized	The given information is not valid.	None
500	Internal Server Error	Something went wrong.	None
503	Service Unavailable	PEM backend database server is down. Service is not available now.	None

36.1.4 Response Schema

Status Code **201**

- Contains information related to when the token was generated and when it expired.*

Name	Type	Required	Restrictions	Description
» issuedAt	string(date-time)	false	none	Token was generated at this time.
» expiresAt	string(date-time)	false	none	Provides date and time when the token expires.
» schema_version	integer	false	none	Application-supported schema version
» backend_schema_version	integer	false	none	Backend database schema version
» version_long	integer	false	none	Version represented in long format
» version	string	false	none	String reprentation of the PEM version
» supported_api_versions	[string]	false	none	List of supported API versions
» deprecated_api_versions	[string]	false	none	List of deprecated API versions

36.1.5 Response Headers

Status	Header	Type	Format	Description
201	X-Subject-Toke	string		Returns the new token, which can be used with REST APIs for further operations

Note

This operation does not require authentication

36.1.6 delete_*token*

Code samples

```
#  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();
}).then(function(body) {
    console.log(body);
});
```

```
DELETE /token/
```

- Release the token.*

Invalidates the token. It will no longer be available for further operations.

36.1.7 Parameters

Name	In	Type	Required	Description
X-Auth-Token	header	string	true	Token of authorization

Example responses

36.1.8 Responses

Status	Meaning	Description	Schema
204	No Content	Successfully released the token from PEM.	
This token can not be used for any further operations.	None		

36.1.9 Response Schema

Note

This operation does not require authentication

EDB Postgres Enterprise Manager REST APIs v16.0

NOTE:

- **EDB Postgres Enterprise Manager** *is referred to as* **PEM** *in this documentation.*

- Because **PEM** *uses self-certified SSL certificates, you may encounter an error such as 'SSL certificate problem: self signed certificate'. You may need to use the 'insecure' connection to connect to the server.*

PEM exposes end-points (REST APIs) for the following operations:

- *Retrieval of the objects retrieved from the monitored server and agents.*
- **C***reate,* **R***etrieve,* **U***pdate, and* **D***elete (CRUD) operations for probes.*
- *Retrieval of the statistics/metrics data (including the historical data) collected by different probes.*
- **C***reate,* **R***etrieve,* **U***pdate, and* **D***elete (CRUD) operations for alerts.*
- **C***reate,* **R***etrieve,* **U***pdate, and* **D***elete (CRUD) operations for the email group used by the alerts mechanism.*
- **C***reate,* **R***etrieve,* **U***pdate, and* **D***elete (CRUD) operations for alert templates used by the alerts.*
- **R***etrieve and* **U***pdate of probe configuration.*
- *Copy the agent/server/database/schema level* **probe configurations** *to other objects at the same level or lower level objects.*
- *Copy the alert configuration of one object to other objects.*
- **R***etrieve and* **U***pdate the PEM-server configurations.*

Base URLs:

- https://PEM-SERVER-IP/pem/api/v16

Email: EnterpriseDB Web: EnterpriseDB License: Limited Use Software License Agreement

Object

Object can be a pem-agent, a monitored server, or a database object (that is, a database, schema, table, index, sequence, function, or view).

get_*server_group*

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/server_group/ \
  -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/v16/server_group/, headers = headers)

print(r.json())
```

```
const headers = {
  Accept:application/json,
  X-Auth-Token:string
```

(continues on next page)

(continued from previous page)

```
};

fetch(https://PEM-SERVER-IP/pem/api/v16/server_group/,
{
  method: GET,

  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

`GET /server_group/`

- Get list of server group.*

Returns a list of server group.

Parameters

Name	In	Type	Required	Description
X-Auth-Token	header	string	true	Token for authentication purpose

Example responses

```
{
  "id": 0,
  "name": "string",
  "hidden": true,
  "parent_id": 0
}
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Returns a list of server group.	*ServerGroupBodyGet*

Note

This operation does not require authentication

post_*server_group*

Code samples

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v16/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/v16/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/v16/server_group/,
{
  method: POST,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

`POST /server_group/`

- Add a new server group.*

Body parameter

```
{
  "name": "string"
}
```

Parameters

Name	In	Type	Required	Description
X-Auth-Token	header	string	true	Token of authorization
body	body	*Server-GroupBody*	true	none

Example responses

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Status 200	None

Response Schema

Note

This operation does not require authentication

get_*server_group*{group_id}

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/server_group/{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/v16/server_group/{group_id}, headers =␣
↪headers)

print(r.json())
```

```
const headers = {
  Accept:application/json,
  X-Auth-Token:string
};
```

(continues on next page)

(continued from previous page)

```
fetch(https://PEM-SERVER-IP/pem/api/v16/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}`

- Get server group by id.*

Returns a server group.

Parameters

Name	In	Type	Required	Description
group_id	path	integer	true	none
X-Auth-Token	header	string	true	Token for authentication purpose

Example responses

```
{
  "id": 0,
  "name": "string",
  "hidden": true,
  "parent_id": 0
}
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	*ServerGroupBodyGet*

Note

This operation does not require authentication

put_*server_group*{group_id}

Code samples

```
#  You can also use wget

curl -X PUT https://PEM-SERVER-IP/pem/api/v16/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/v16/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/v16/server_group/{group_id},
{
  method: PUT,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

`PUT /server_group/{group_id}`

- Update a server group.*

Body parameter

```
{
  "name": "string"
}
```

Parameters

Name	In	Type	Required	Description
group_id	path	integer	true	none
X-Auth-Token	header	string	true	Token of authorization
body	body	*Server-GroupBody*	true	Updated successfully

Example responses

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Status 200	None

Response Schema

Note

This operation does not require authentication

delete_*server_group*{group_id}

Code samples

```
#  You can also use wget

curl -X DELETE https://PEM-SERVER-IP/pem/api/v16/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/v16/server_group/{group_id},␣
↪headers = headers)

print(r.json())
```

```
const headers = {
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v16/server_group/{group_id},
{
```

(continues on next page)

(continued from previous page)

```
  method: DELETE,

  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

```
DELETE /server_group/{group_id}
```

- Delete a server group.*

Parameters

Name	In	Type	Required	Description
group_id	path	integer	true	none
X-Auth-Token	header	string	true	Token for authentication purpose

Example responses

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Status 200	None

Response Schema

> **Note**
>
> This operation does not require authentication

get_*agent*

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/agent/ \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
```

(continues on next page)

(continued from previous page)

```
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v16/agent/, headers = headers)

print(r.json())
```

```
const headers = {
  Accept:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v16/agent/,
{
  method: GET,

  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

`GET /agent/`

- Retrieve the pem-agent information.*

Returns a list of information and status for the pem-agents visible to the user (represented by the token).

Parameters

Name	In	Type	Required	Description
X-Auth-Token	header	string	true	Token for authentication purpose

Example responses

```
[
  {
    "id": 0,
    "name": "string",
    "version": "string",
    "profile_id": 0,
    "active": true,
    "heartbeat_tolerance": 0,
    "alert_blackout": true,
    "owner": "string",
    "team": "string",
    "gid": 0,
    "tags": [
```

(continues on next page)

(continued from previous page)

```
      "string"
    ],
    "ignore_mnt_points": [
      "string"
    ]
  }
]
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Returns a list of information and status about the pem-agents.	*Certificates and key files for PEM agents*

Note

This operation does not require authentication

get_*agent*{agent_id}

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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/v16/agent/{agent_id}, headers =␣
↪headers)

print(r.json())
```

```
const headers = {
  Accept:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v16/agent/{agent_id},
{
```

(continues on next page)

(continued from previous page)

```
  method: GET,

  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

`GET /agent/{agent_id}`

- Retrieve information for an agent, identified by the agent_id.*

Parameters

Name	In	Type	Required	Description
agent_id	path	integer	true	none
X-Auth-Token	header	string	true	Token of authorization

Example responses

```
{
  "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"
  ]
}
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	*PEM backend database server and agent connection certificates*

> **Note**
>
> This operation does not require authentication

put_*agent*{agent_id}

Code samples

```
#  You can also use wget

curl -X PUT https://PEM-SERVER-IP/pem/api/v16/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.put(https://PEM-SERVER-IP/pem/api/v16/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"
      }
    ]
```

(continues on next page)

(continued from previous page)

```
  },
  "ignore_mnt_points": [
    "string"
  ]
};
const headers = {
  Content-Type:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v16/agent/{agent_id},
{
  method: PUT,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

`PUT /agent/{agent_id}`

- Update agent information.*

Body parameter

```
{
  "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"
```

(continues on next page)

(continued from previous page)

```
      }
    ]
  },
  "ignore_mnt_points": [
    "string"
  ]
}
```

Parameters

Name	In	Type	Required	Description
agent_id	path	integer	true	none
X-Auth-Token	header	string	true	Token of authorization
body	body	*AgentBody*	true	none

Example responses

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Updated successfully	None

Response Schema

Note

This operation does not require authentication

delete_*agent*{agent_id}

Code samples

```
#  You can also use wget

curl -X DELETE https://PEM-SERVER-IP/pem/api/v16/agent/{agent_id} \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  X-Auth-Token: string
}

r = requests.delete(https://PEM-SERVER-IP/pem/api/v16/agent/{agent_id}, headers =␣
↪headers)
```

(continues on next page)

(continued from previous page)

```
print(r.json())
```

```
const headers = {
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v16/agent/{agent_id},
{
  method: DELETE,

  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

`DELETE /agent/{agent_id}`

- Stop managing the agent by PEM.*

Parameters

Name	In	Type	Required	Description
agent_id	path	integer	true	none
X-Auth-Token	header	string	true	Token of authorization

Example responses

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Status 200	None

Response Schema

Note

This operation does not require authentication

get_*server*

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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/v16/server/, headers = headers)

print(r.json())
```

```
const headers = {
  Accept:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v16/server/,
{
  method: GET,

  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

`GET /server/`

- Retrieval of the managed server list*

Parameters

Name	In	Type	Required	Description
X-Auth-Token	header	string	true	Token of authorization

Example responses

```
[
  {
    "id": 0,
    "name": "string",
    "host": "string",
    "port": 0,
    "db": "string",
```

(continues on next page)

(continued from previous page)


```
    "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"
    ],
    "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"
```


(continues on next page)

(continued from previous page)

```
  }
]
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Status 200	Inline

Response Schema

Status Code **200**

Name	Type	Required	Restrictions	Description
anonymous	*Query information*]	false	none	none
» id	integer	false	none	Server ID.NOTE: This property is not applicable for a POST request.
» name	string	false	none	Server name
» host	string	false	none	Server host
» port	integer	false	none	Server port
» db	string	false	none	Maintenance databse
» profile_id	integer	false	none	Profile ID
» serviceid	string	false	none	Server service ID
» connection_params	[object]	false	none	List of connection parameters for the server.
»» name	string	false	none	Parameter name
»» keyword	string	false	none	Parameter keyword
»» value	any	false	none	Parameter value
» active	boolean	false	none	Determines whether the server is active.
» alert_blackout	boolean	false	none	Determines whether the alerts on the server are blacked out.
» owner	integer	false	none	ID of the owner who registered the server with PEM.
» team	string	false	none	Defines the role/username who can see server information.
» server_owner	string	false	none	Name of the user who registered the server on the PEM database server.

continues on next page

Table 27 – continued from previous page

Name	Type	Required	Restrictions	Description
» is_remote_monitoring	boolean	false	none	Determines whether this server is being monitored remotely (that is, the monitoring pemAgent does not reside on the same host as the monitored server).
» tags	[string]	false	none	List of tags associated with the server.
» replication_solution	string	false	none	Replication solution used by the server
» efm_cluster_name	string	false	none	Cluster name of EFM
» efm_service_name	string	false	none	Service name of EFM
» efm_installation_path	string	false	none	Installation path of EFM
» patroni_cluster_name	string	false	none	Cluster name of Patroni
» patroni_installation_path	string	false	none	Installation path of Patroni
» patroni_config_path	string	false	none	Configuration path of Patroni
» comment	string	false	none	Description of the server
» username	string	false	none	Username for connecting the database server
» gid	integer	false	none	ID of the server group
» server_group_name	string	false	none	Name of the server group
» db_res	string	false	none	Database restriction SQL clause
» role	string	false	none	The role the connection will use
» is_edb	boolean	false	none	Is server an EDB Advanced Server?
» agent_id	integer	false	none	Agent ID bound to the server
» asb_host	string	false	none	PEM database host used by agent while creating PEM connection
» asb_port	integer	false	none	PEM database port used by agent while creating PEM connection
» asb_username	string	false	none	PEM database user name used by agent while creating PEM connection
» asb_database	string	false	none	PEM database to which agent will connect
» asb_sslmode	string	false	none	PEM database SSL mode
» agent_allowtakeover	boolean	false	none	Specifies whether the server can be taken over by another agent.
» agent_capability_list	[string]	false	none	Array of agent capabilities
» agent_description	string	false	none	Agent description

continues on next page

Table 27 – continued from previous page

Name	Type	Required	Restrictions	Description
» sslcompression	boolean	false	none	Determines whether data sent over SSL should be compressed.
» use_ssh_tunnel	boolean	false	none	Determines whether to use ssh tunnel.
» tunnel_host	string	false	none	Tunnel host name
» tunnel_port	integer	false	none	Tunnel port number
» tunnel_username	string	false	none	Tunnel username
» tunnel_authentication	boolean	false	none	Determines whether to use tunnel authentication.
» tunnel_identity_file	string	false	none	Tunnel identity file used for authentication.
» tunnel_keep_alive	integer	false	none	Interval in seconds to send keep alive packets for ssh tunnel.
» post_connection_sql	string	false	none	SQL to be executed after establishing the connection.

Enumerated Values

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

Note

This operation does not require authentication

post_*server*

Code samples

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v16/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/v16/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,
```

(continues on next page)

(continued from previous page)

```
  "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"
};
const headers = {
  Content-Type:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v16/server/,
{
  method: POST,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

`POST /server/`

- Add a new managed database server.*

Body parameter

```
{
  "name": "string",
  "host": "string",
  "port": 0,
  "db": "string",
  "profile_id": 0,
```

(continues on next page)

(continued from previous page)


```
  "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",
  "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",
```


(continues on next page)

(continued from previous page)

```
  "tunnel_port": 22,
  "tunnel_username": "string",
  "tunnel_password": "string",
  "tunnel_authentication": false,
  "tunnel_identity_file": "string",
  "tunnel_keep_alive": 0,
  "post_connection_sql": "string"
}
```

Parameters

Name	In	Type	Required	Description
X-Auth-Token	header	string	true	Token of authorization
body	body	*ServerBody*	true	none

Example responses

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Status 200	None

Response Schema

Note

This operation does not require authentication

get_*server*{server_id}

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/server/{server_id} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}
```

(continues on next page)

(continued from previous page)

```
r = requests.get(https://PEM-SERVER-IP/pem/api/v16/server/{server_id}, headers =␣
↪headers)

print(r.json())
```

```
const headers = {
  Accept:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v16/server/{server_id},
{
  method: GET,

  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

`GET /server/{server_id}`

- Retrieve the information for the server, identified by the server_id.*

Parameters

Name	In	Type	Required	Description
server_id	path	string	true	none
X-Auth-Token	header	string	true	Token of authorization

Example responses

```
{
  "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,
```

(continues on next page)

(continued from previous page)


```
  "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"
  ],
  "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"
}
```


Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Created successfully	*Web server certificates*

Note

This operation does not require authentication

put_*server*{server_id}

Code samples

```
#  You can also use wget

curl -X PUT https://PEM-SERVER-IP/pem/api/v16/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.put(https://PEM-SERVER-IP/pem/api/v16/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"
```

(continues on next page)

(continued from previous page)

```
      }
    ],
    "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",
  "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",
```

(continues on next page)

(continued from previous page)

```
  "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/v16/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}`

- Update the server information.*

Body parameter

```
{
  "name": "string",
  "host": "string",
  "port": 0,
  "db": "string",
  "profile_id": 0,
  "serviceid": "string",
  "connection_params": {
    "added": [
      {
        "keyword": "hostaddr",
        "name": "hostaddr",
        "value": "qwde"
      }
```

(continues on next page)

(continued from previous page)


```
    ],
    "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": [
      {
        "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",
```


(continues on next page)

(continued from previous page)


```
  "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"
}
```


Parameters

Name	In	Type	Required	Description
server_id	path	string	true	none
X-Auth-Token	header	string	true	Token of authorization
body	body	*Server-BodyUpdate*	true	none

Example responses

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Updated successfully.	None

Response Schema

Note

This operation does not require authentication

delete_*server*{server_id}

Code samples

```
#  You can also use wget

curl -X DELETE https://PEM-SERVER-IP/pem/api/v16/server/{server_id} \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  X-Auth-Token: string
}

r = requests.delete(https://PEM-SERVER-IP/pem/api/v16/server/{server_id}, headers =␣
↪headers)

print(r.json())
```

```
const headers = {
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v16/server/{server_id},
{
  method: DELETE,

  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

`DELETE /server/{server_id}`

- Stop managing the server by PEM.*

Parameters

Name	In	Type	Required	Description
server_id	path	string	true	none
X-Auth-Token	header	string	true	Token of authorization

Example responses

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Status 200	None

Response Schema

> **Note**
>
> This operation does not require authentication

get_*server*{sid}*database*

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/server/{sid}/database/ \
  -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/v16/server/{sid}/database/, headers =␣
↪headers)

print(r.json())
```

```
const headers = {
  Accept:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v16/server/{sid}/database/,
{
  method: GET,

  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

`GET /server/{sid}/database/`

- Retrieve the list of properties of database(s) for a managed server.*

Parameters

Name	In	Type	Required	Description
X-Auth-Token	header	string	true	Token of authorization
sid	path	integer	true	Server ID

Example responses


```
[
  {
    "database_name": "string",
    "connections_allowed": true,
    "system_database": true,
    "recorded_time": "2019-08-24T14:15:22Z",
    "encoding": "string",
    "server_id": 0
  }
]
```


Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Status 200	Inline

Response Schema

Status Code **200**

Name	Type	Required	Restrictions	Description
anonymous	*Query information*]	false	none	none
» database_name	string	false	none	Database name
» connections_allowed	boolean	false	none	Determines whether a connection to the database is allowed.
» system_database	boolean	false	none	Determines whether the database is a system database.
» recorded_time	string(date-time)	false	none	Specifies when the database was first detected by the PEM agent.
» encoding	string	false	none	Database encoding
» server_id	integer	false	none	Server ID of database

> **Note**
>
> This operation does not require authentication

get_*server*{sid}*database*{db_name}

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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/v16/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/v16/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}`

- Retrieve the properties of a particular database of a managed server.*

Parameters

Name	In	Type	Required	Description
sid	path	integer	true	Server ID

continues on next page

Table 40 – continued from previous page

Name	In	Type	Required	Description
db_name	path	string	true	none
X-Auth-Token	header	string	true	Token of authorization

Example responses

```
{
  "database_name": "string",
  "connections_allowed": true,
  "system_database": true,
  "recorded_time": "2019-08-24T14:15:22Z",
  "encoding": "string",
  "server_id": 0
}
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Status 200	*PEM backend database server and agent connection certificates*

Note

This operation does not require authentication

get_*server*{sid}*database*{db_name}*schema*

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v16/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/v16/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/`

- Retrieve the list of properties of schema(s) for a particular database of a managed server.*

Parameters

Name	In	Type	Required	Description
sid	path	integer	true	Server ID
db_name	path	string	true	Database name
X-Auth-Token	header	string	true	Token of authorization

Example responses

```
[
  {
    "schema_name": "string",
    "database_name": "string",
    "server_id": 0,
    "recorded_time": "2019-08-24T14:15:22Z"
  }
]
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Status 200	Inline

Response Schema

Status Code **200**

Name	Type	Required	Restrictions	Description
anonymous	*Query information*]	false	none	none
» schema_name	string	false	none	Schema name
» database_name	string	false	none	Database name of schema
» server_id	integer	false	none	Server ID schema
» recorded_time	string(date-time)	false	none	Specifies when schema was first detected by PEM agent.

Note

This operation does not require authentication

get_*server*{sid}*database*{db_name}*schema*{schema_name}

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/server/{sid}/database/{db_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/v16/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/v16/server/{sid}/database/{db_name}/schema/
↪{schema_name},
{
```

(continues on next page)

(continued from previous page)

```
  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}`

- Retrieve the properties of schema(s) for a particular database of a managed server.*

Parameters

Name	In	Type	Required	Description
sid	path	integer	true	Server ID
db_name	path	string	true	Database name
schema_name	path	string	true	Schema name
X-Auth-Token	header	string	true	Token of authorization

Example responses

```
{
  "schema_name": "string",
  "database_name": "string",
  "server_id": 0,
  "recorded_time": "2019-08-24T14:15:22Z"
}
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Status 200	*Using the Schema Diff tool*

Note

This operation does not require authentication

get_*server*{sid}*database*{db_name}*schema*{schema_name}*table*

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v16/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/v16/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/`

- Retrieve the list of properties of tables residing in a schema for a particular database of a managed server.*

Parameters

Name	In	Type	Required	Description
sid	path	integer	true	Server ID
db_name	path	string	true	Database name
schema_name	path	string	true	Schema name
X-Auth-Token	header	string	true	Token of authorization

Example responses

```
[
  {
    "table_name": "string",
    "schema_name": "string",
    "database_name": "string",
    "server_id": 0,
    "has_primary_key": true,
    "recorded_time": "string"
  }
]
```


Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Status 200	Inline

Response Schema

Status Code **200**

Name	Type	Required	Restrictions	Description
anonymous	*Query information*]	false	none	none
» table_name	string	false	none	Table name
» schema_name	string	false	none	Schema name
» database_name	string	false	none	Database name
» server_id	integer	false	none	Server ID
» has_primary_key	boolean	false	none	Determines whether table has a primary key.
» recorded_time	string	false	none	Specifies when table was first detected by the PEM agent.

Note

This operation does not require authentication

get_*server*{sid}*database*{db_name}*schema*{schema_name}*table*{table_name}

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/server/{sid}/database/{db_name}/schema/
↪{schema_name}/table/{table_name} \
```

(continues on next page)

(continued from previous page)

```
  -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/v16/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/v16/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}`

- Retrieve the properties of tables residing in a schema for a particular database of a managed server.*

Parameters

Name	In	Type	Required	Description
sid	path	integer	true	Server ID
db_name	path	string	true	Database name
schema_name	path	string	true	Schema name
table_name	path	string	true	none
X-Auth-Token	header	string	true	Token of authorization

Example responses

```
{
  "table_name": "string",
  "schema_name": "string",
```

(continues on next page)

(continued from previous page)

```
  "database_name": "string",
  "server_id": 0,
  "has_primary_key": true,
  "recorded_time": "string"
}
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Status 200	*Table relationship options*

Note

This operation does not require authentication

get_*server*{sid}*database*{db_name}*schema*{schema_name}*index*

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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/v16/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/v16/server/{sid}/database/{db_name}/schema/
↪{schema_name}/index/,
{
  method: GET,
```

(continues on next page)

(continued from previous page)

```
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

`GET /server/{sid}/database/{db_name}/schema/{schema_name}/index/`

- Retrieve the list of properties of indexes residing in a schema for a particular database of a managed server.*

Parameters

Name	In	Type	Required	Description
sid	path	integer	true	Server ID
db_name	path	string	true	Database name
schema_name	path	string	true	Schema name
X-Auth-Token	header	string	true	Token of authorization

Example responses

```
[
  {
    "index_name": "string",
    "ind_keys": [
      0
    ],
    "table_name": "string",
    "schema_name": "string",
    "database_name": "string",
    "server_id": 0,
    "recorded_time": "string"
  }
]
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Status 200	Inline

Response Schema

Status Code **200**

Name	Type	Required	Restrictions	Description
anonymous	*Query information*]	false	none	none
» index_name	string	false	none	Index name
» ind_keys	[integer]	false	none	Indexed column positions in table
» table_name	string	false	none	Table name
» schema_name	string	false	none	Schema
» database_name	string	false	none	Database name
» server_id	integer	false	none	Server ID
» recorded_time	string	false	none	Specifies when index was first detected by PEM agent.

Note

This operation does not require authentication

get_*server*{sid}*database*{db_name}*schema*{schema_name}*index*{index_name}

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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/v16/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/v16/server/{sid}/database/{db_name}/schema/
↪{schema_name}/index/{index_name},
{
  method: GET,
```

(continues on next page)

(continued from previous page)

```
  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}`

- Retrieve the properties of indexes residing in a schema for a particular database of a managed server.*

Parameters

Name	In	Type	Required	Description
sid	path	integer	true	Server ID
db_name	path	string	true	Database name
schema_name	path	string	true	Schema name
index_name	path	string	true	none
X-Auth-Token	header	string	true	Token of authorization

Example responses

```
{
  "index_name": "string",
  "ind_keys": [
    0
  ],
  "table_name": "string",
  "schema_name": "string",
  "database_name": "string",
  "server_id": 0,
  "recorded_time": "string"
}
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Status 200	get__alert_config_server_{server_id}_database_{database_name}_schema_{index_name}

Note

This operation does not require authentication

get_*server*{sid}*database*{db_name}*schema*{schema_name}*sequence*

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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/v16/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/v16/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/`

- Retrieve the list of properties of sequences residing in a schema for a particular database of a managed server.*

Parameters

Name	In	Type	Required	Description
sid	path	integer	true	Server ID
db_name	path	string	true	Database name
schema_name	path	string	true	Schema name
X-Auth-Token	header	string	true	Token of authorization

Example responses


```
[
  {
    "sequence_name": "string",
    "schema_name": "string",
    "database_name": "string",
    "server_id": 0,
    "recorded_time": "2019-08-24T14:15:22Z"
  }
]
```


Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Status 200	Inline

Response Schema

Status Code **200**

Name	Type	Required	Restrictions	Description
anonymous	*Query information*]	false	none	none
» sequence_name	string	false	none	Sequence name
» schema_name	string	false	none	Schema name
» database_name	string	false	none	Database name
» server_id	integer	false	none	Server ID
» recorded_time	string(date-time)	false	none	Specifies when sequence was first detected by PEM agent.

Note

This operation does not require authentication

get_*server*{sid}*database*{db_name}*schema*{schema_name}*sequence*{sequence_name}

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/server/{sid}/database/{db_name}/schema/
↪{schema_name}/sequence/{sequence_name} \
```

(continues on next page)

(continued from previous page)

```
  -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/v16/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/v16/server/{sid}/database/{db_name}/schema/
↪{schema_name}/sequence/{sequence_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}/sequence/{sequence_name}`

- Retrieve the properties of sequences residing in a schema for a particular database of a managed server.*

Parameters

Name	In	Type	Required	Description
sid	path	integer	true	Server ID
db_name	path	string	true	Database name
schema_name	path	string	true	Schema name
sequence_name	path	string	true	none
X-Auth-Token	header	string	true	Token of authorization

Example responses

```
{
  "sequence_name": "string",
  "schema_name": "string",
```

(continues on next page)

(continued from previous page)

```
  "database_name": "string",
  "server_id": 0,
  "recorded_time": "2019-08-24T14:15:22Z"
}
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Status 200	get__alert_config_server_{server_id}_database_{database_name}_sequence_{sequence_name}

Note

This operation does not require authentication

get_*server*{sid}*database*{db_name}*schema*{schema_name}*view*

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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/v16/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/v16/server/{sid}/database/{db_name}/schema/
↪{schema_name}/view/,
{
  method: GET,
```

(continues on next page)

(continued from previous page)

```
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

`GET /server/{sid}/database/{db_name}/schema/{schema_name}/view/`

- Retrieve the list of properties of views residing in a schema for a particular database of a managed server.*

Parameters

Name	In	Type	Required	Description
sid	path	integer	true	Server ID
db_name	path	string	true	Database name
schema_name	path	string	true	Schema name
X-Auth-Token	header	string	true	Token of authorization

Example responses

```
[
  {
    "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"
  }
]
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Status 200	Inline

Response Schema

Status Code **200**

Name	Type	Required	Restrictions	Description
anonymous	*Query information*]	false	none	none
» view_name	string	false	none	View name
» view_type	string	false	none	View type
» ispopulated	boolean	false	none	Determines whether the view is populated.
» view_owner	string	false	none	View owner
» definition	string	false	none	View definition
» tablespace_name	string	false	none	View tablespace
» schema_name	string	false	none	View schema
» database_name	string	false	none	View database
» server_id	integer	false	none	View server ID
» recorded_time	string	false	none	Specifies when view was first detected by PEM agent.

Note

This operation does not require authentication

get_*server*{sid}*database*{db_name}*schema*{schema_name}*view*{view_name}

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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/v16/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/v16/server/{sid}/database/{db_name}/schema/
↪{schema_name}/view/{view_name},
```

(continues on next page)

(continued from previous page)

```
{
  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}`

- Retrieve the properties of views residing in a schema for a particular database of a managed server.*

Parameters

Name	In	Type	Required	Description
sid	path	integer	true	Server ID
db_name	path	string	true	Database name
schema_name	path	string	true	Schema name
view_name	path	string	true	none
X-Auth-Token	header	string	true	Token of authorization

Example responses

```
{
  "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"
}
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Status 200	*Reviewing or modifying agent properties*

> **Note**
>
> This operation does not require authentication

get_*server*{sid}*database*{db_name}*schema*{schema_name}*function*

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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/v16/server/{sid}/database/{db_name}/
↪schema/{schema_name}/function/, headers = headers)

print(r.json())
```

```
const headers = {
  Accept:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v16/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/`

- Retrieve the list of properties of functions residing in a schema for a particular database of a managed server.*

Parameters

Name	In	Type	Required	Description
sid	path	integer	true	Server ID
db_name	path	string	true	Database name
schema_name	path	string	true	Schema name
X-Auth-Token	header	string	true	Token of authorization

Example responses


```
[
  {
    "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"
  }
]
```


Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Status 200	Inline

Response Schema

Status Code **200**

Name	Type	Required	Restrictions	Description
anonymous	*Query information*]	false	none	none
» function_name	string	false	none	Function name
» function_type	string	false	none	Function type
» return_type	string	false	none	Function return type
» arg_types	[string]	false	none	Function argument types
» function_binary	string	false	none	none
» extension_name	string	false	none	none

continues on next page

Table 69 – continued from previous page

Name	Type	Required	Restrictions	Description
» package_name	string	false	none	Function package name
» schema_name	string	false	none	Function schema
» database_name	string	false	none	Function database name
» server_id	integer	false	none	Function server id
» recorded_time	string(date-time)	false	none	Specifies when function was first detected by PEM agent.

Note

This operation does not require authentication

get_*server*{sid}*database*{db_name}*schema*{schema_name}*function*{function_name}

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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/v16/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/v16/server/{sid}/database/{db_name}/schema/
↪{schema_name}/function/{function_name},
{
  method: GET,

  headers: headers
})
.then(function(res) {
    return res.json();
```

(continues on next page)

(continued from previous page)

```
}).then(function(body) {
    console.log(body);
});
```

```
GET /server/{sid}/database/{db_name}/schema/{schema_name}/function/{function_name}
```

- Retrieve the properties of functions residing in a schema for a particular database of a managed server.*

Parameters

Name	In	Type	Required	Description
sid	path	integer	true	Server ID
db_name	path	string	true	Database name
schema_name	path	string	true	Schema name
function_name	path	string	true	none
X-Auth-Token	header	string	true	Token of authorization

Example responses

```
{
  "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"
}
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Status 200	*Functionality affected by limiting operating system privileges*

Note

This operation does not require authentication

Profiles

Manage Profiles API: create, list, update, publish/rollback drafts, and reassign/delete profiles for servers and agents.

get_*profiles*

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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/v16/profiles/, headers = headers)

print(r.json())
```

```
const headers = {
  Accept:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v16/profiles/,
{
  method: GET,

  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

`GET /profiles/`

- List profiles*

Returns a summary list of published profiles only. If a draft exists for a published profile, the item shows that profile's ID and 'published' is false. The response does not include 'target_probe_configs' or 'target_alert_configs'.

Parameters

Name	In	Type	Required	Description
X-Auth-Token	header	string	true	Token for authentication purpose

Example responses


```
[
  {
    "id": 34,
    "name": "Production Agents",
    "description": "Monitoring production agents",
    "target_kind": "a",
    "active_assignments": 0,
    "published": true
  }
]
```


Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	Inline

Response Schema

Status Code **200**

Name	Type	Required	Restrictions	Description
anonymous	*Query information*]	false	none	[A trimmed profile representation for list operations (no probe/alert config arrays).]
» id	integer	false	none	none
» name	string	false	none	none
» description	string	false	none	none
» target_kind	string	false	none	Target type for the profile: 's' for server, 'a' for agent.
» published	boolean	false	none	Convenience flag: true when status is 'published'.
» active_assignments	integer	false	none	Count of active servers/agents assigned to this profile.

Enumerated Values

Property	Value
target_kind	s
target_kind	a

> **Note**
>
> This operation does not require authentication

post_*profiles*

Code samples

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v16/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/v16/profiles/, headers = headers)

print(r.json())
```

```
const inputBody = {
  "name": "Production Servers",
  "description": "Monitoring production database servers",
  "target_kind": "s"
};
const headers = {
  Content-Type:application/json,
  Accept:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v16/profiles/,
{
  method: POST,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
```

(continues on next page)

(continued from previous page)

```
    console.log(body);
});
```

`POST /profiles/`

- Create a new profile*

Creates a new profile with the specified name (required), description (optional), and target_kind (optional, defaults to 's' for server). Probes and auto-created alerts will be automatically created based on target_kind.

Body parameter

```
{
  "name": "Production Servers",
  "description": "Monitoring production database servers",
  "target_kind": "s"
}
```

Parameters

Name	In	Type	Required	Description
X-Auth-Token	header	string	true	Token for authorization
body	body	object	false	none
» name	body	string	true	Profile name (required)
» description	body	string	false	Profile description (optional)
» target_kind	body	string	false	Target kind: 's' for server, 'a' for agent (optional, defaults to 's')

Enumerated Values

Parameter	Value
» target_kind	s
» target_kind	a

Example responses

```
{
  "success": 0,
  "data": "string"
}
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Profile created successfully	Inline
400	Bad Request	Bad request - missing required field 'name'	None
500	Internal Server Error	Internal server error	None

Response Schema

Status Code **200**

Name	Type	Required	Restrictions	Description
» success	integer	false	none	none
» data	string	false	none	none

Note

This operation does not require authentication

get_*profiles*{profile_id}

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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/v16/profiles/{profile_id}, headers =␣
↪headers)

print(r.json())
```

```
const headers = {
  Accept:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v16/profiles/{profile_id},
{
```

(continues on next page)

(continued from previous page)

```
  method: GET,

  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

`GET /profiles/{profile_id}`

- Get a profile by id*

Returns a single profile with full probe and alert configurations. The response includes a boolean 'published' field (true when the profile is published).

Parameters

Name	In	Type	Required	Description
profile_id	path	integer	true	none
X-Auth-Token	header	string	true	Token for authorization

Example responses


```
{
  "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",
```


(continues on next page)

(continued from previous page)


```
    "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": [
```


(continues on next page)

(continued from previous page)

```
          0
        ]
      }
  ]
}
```

Profile not found

```
{
  "message": "Not found"
}
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	*Upgrading SQL Profiler*
404	Not Found	Profile not found	Inline

Response Schema

Status Code **404**

Name	Type	Required	Restrictions	Description
» message	string	false	none	none

Note

This operation does not require authentication

put_*profiles*{profile_id}

Code samples

```
#  You can also use wget

curl -X PUT https://PEM-SERVER-IP/pem/api/v16/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,
```

(continues on next page)

(continued from previous page)

```
  X-Auth-Token: string
}

r = requests.put(https://PEM-SERVER-IP/pem/api/v16/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
      }
```

(continues on next page)

(continued from previous page)

```
    ],
    "deleted": [
      {
        "id": 103
      }
    ]
  }
};
const headers = {
  Content-Type:application/json,
  Accept:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v16/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}`

- Update a profile by id*

Update a single profile identified by 'profile_id'. Use 'name'/'description' for metadata. For probes, use 'target_probe_configs.changed' to modify existing probe configs only (keys: probe_id, enabled?, enabled_by_default?, execution_frequency?, lifetime?; also accepts legacy 'use_default_enabled' and 'interval'). Adding probes via this endpoint is not supported. For alerts, use 'target_alert_configs' with changed/added/deleted (deleted accepts objects with only 'id').

Body parameter

```
{
  "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
      }
```

(continues on next page)

(continued from previous page)

```
    ]
  },
  "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
      }
    ]
  }
}
```

Parameters

Name	In	Type	Required	Description
profile_id	path	integer	true	none
X-Auth-Token	header	string	true	Token for authorization
body	body	object	true	none
» name	body	string	false	New name for the profile (optional).
» description	body	string	false	New description/notes for the profile (optional).

continues on next page

Table 83 – continued from previous page

Name	In	Type	Required	Description
» target_probe_configs	body	*ProbeConfigsUpdateOnly*	false	Delta for probe configs when updating by profile id: only modify existing probes using 'changed'. Adding probes is not supported via this endpoint.
»» changed	body	*Query information*]	false	Existing probe configs to modify (identified by probe_id).
»»» probe_id	body	integer	true	Probe identifier to add/update.
»»» enabled	body	boolean	false	Enable/disable this probe in the profile.
»»» enabled_by_default	body	boolean	false	Whether the probe follows default enablement. Alias: 'use_default_enabled' accepted.
»»» execution_frequency	body	integer	false	Probe execution frequency in minutes. Alias: 'interval' accepted.
»»» lifetime	body	integer	false	History retention (days).
» target_alert_configs	body	*AlertConfigsDelta*	false	Delta for alert configs within a profile.
»» changed	body	[allOf]	false	Modify existing alerts.
»»» id	body	integer	false	Alert ID to modify.
»»» name	body	string	false	Alert name.
»»» enabled	body	boolean	false	none
»»» history_retention	body	integer	false	none
»»» frequency_min	body	integer	false	none
»»» operator	body	string	false	none
»»» low_threshold_value	body	number	false	none
»»» medium_threshold_value	body	number	false	none
»»» high_threshold_value	body	number	false	none
»»» all_alert_enable	body	boolean	false	none
»»» email_group_id	body	integer	false	none
»»» low_alert_enable	body	boolean	false	none
»»» low_email_group_id	body	integer	false	none
»»» med_alert_enable	body	boolean	false	none
»»» med_email_group_id	body	integer	false	none
»»» high_alert_enable	body	boolean	false	none
»»» high_email_group_id	body	integer	false	none
»» added	body	*Query information*]	false	Add new alerts. See AlertPayload for required fields.
»»» alert_name	body	string	false	Name of the alert. NOTE: This property is a mandatory parameter for a POST request.

continues on next page

Table 83 – continued from previous page

Name	In	Type	Required	Description
»»»» alert_template	body	integer	false	ID of the alert template. The ID must be a positive integer value. NOTE: This property is a mandatory parameter for a POST request.
»»»» description	body	string	false	none
»»»» enabled	body	boolean	false	Defines whether this alert is enabled/disabled. NOTE: This property is a mandatory parameter for a POST request.
»»»» history_retention	body	integer	false	Specifies how many days to keep the history of this alert. Possible values are from 1 to 99999. NOTE: This property is a mandatory parameter for a POST request.
»»»» frequency_min	body	integer	false	Defines how frequently (in minutes) the alert must be checked on PEM database server. Possible values are 1 to 65534. NOTE: This property is a mandatory parameter for a POST request.
»»»» operator	body	string	false	Defines the operator for checking threshold values. Valid string are ">" and "<". NOTE: This property is a mandatory parameter for the POST request.
»»»» low_threshold_value	body	number	false	Threshold value for triggering the low alert. This property can be any valid integer/float value. NOTE: It is a mandatory parameter for a POST request.
»»»» medium_threshold_value	body	number	false	Threshold value for triggering the medium alert. This property can be any valid integer/float value. NOTE: It is a mandatory parameter for a POST request.

continues on next page

Table 83 – continued from previous page

Name	In	Type	Required	Description
»»»» high_threshold_value	body	number	false	Threshold value for triggering the high alert. This property can be any valid integer/float value. NOTE: It is a mandatory parameter for a POST request.
»»»» all_alert_enable	body	boolean	false	Determines whether to send an email notification to the group specified by the parameter 'email_group_id' when the state of the alert changes.
»»»» email_group_id	body	integer	false	Defines the email group to which email notification are sent, when the state of the alert changes.
»»»» low_alert_enable	body	boolean	false	Determines whether to send an email notification to the group specified by the parameter 'low_email_group_id' when the state of the alert is set to LOW.
»»»» low_email_group_id	body	integer	false	Defines the email group to which email notifications are sent when the state of the alert changes to LOW.
»»»» med_alert_enable	body	boolean	false	Determines whether to send an email notification to the group specified by the parameter 'medium_email_group_id' when the state of the alert is set to MEDIUM.
»»»» med_email_group_id	body	integer	false	Defines the email group to which email notifications are sent when the state of the alert changes to MEDIUM.
»»»» high_alert_enable	body	boolean	false	Determines whether to send an email notification to the group specified by the parameter 'high_email_group_id' when the state of the alert is set to HIGH.

continues on next page

Table 83 – continued from previous page

Name	In	Type	Required	Description
»»»» high_email_group_id	body	integer	false	Defines the email group to which email notification are sent when the state of the alert changes to HIGH.
»»»» send_trap	body	boolean	false	Determines whether to send an SNMP trap notification when the state of the alert changes.
»»»» snmp_trap_version	body	integer	false	Defines the SNMP trap version. Supported SNMP trap versions are 1, 2 or 3.
»»»» low_send_trap	body	boolean	false	Determines whether to send SNMP trap notifications when the state of the alert changes to LOW.
»»»» med_send_trap	body	boolean	false	Determines whether to send SNMP trap notifications when the state of the alert changes to MEDIUM.
»»»» high_send_trap	body	boolean	false	Determines whether to send SNMP trap notifications when the state of the alert chagnes to HIGH.
»»»» cleared_alert_enable	body	boolean	false	Determines whether to mute the notification for 'clear alert'.
»»»» params	body	*Query information*]	false	List of parameter/value pairs to be used by the alert template for checking alerts. NOTE: The alert template requires the all parameter to be defined.
»»»»» paramname	body	string	false	Name of the parameter. NOTE: This property must present on the alert template parameter list.Example:
»»»»» paramvalue	body	string	false	Value for the given parameter
»»»» execute_script	body	boolean	false	Determines whether to execute the provided script if the state of the alert changes to LOW/MEDIUM/HIGH.
»»»» execute_script_on_clear	body	boolean	false	Determines whether to execute the provided script if the state of the alert clears.

continues on next page

Table 83 – continued from previous page

Name	In	Type	Required	Description
»»»» execute_script_on_pem_server	body	boolean	false	Determines whether to execute the provided script on the PEM host or the host where the agent is monitoring the object.
»»»» script_code	body	string	false	The shell or batch script to be executed. Please read the documentation for detailed information.
»»»» submit_to_nagios	body	boolean	false	Defines whether to send notification to the nagios, when state of the alert is chagned.
»»»» override_default_config	body	boolean	false	Determines whether to override the default webhook configuration.
»»»» send_notification	body	boolean	false	Determines whether the webhook send notification is enabled.
»»»» low_webhook_ids	body	[integer]	false	Defines webhook IDs for which low alerts can be configured.
»»»» med_webhook_ids	body	[integer]	false	Defines webhook IDs for which medium alerts can be configured.
»»»» high_webhook_ids	body	[integer]	false	Defines webhook IDs for which high alerts can be configured.
»»»» cleared_webhook_ids	body	[integer]	false	Defines webhook IDs for which cleared alerts can be configured.
»»» deleted	body	*Query information*]	false	Delete existing alerts by id.
»»»» id	body	integer	true	Alert config id to delete.

Enumerated Values

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%)

Example responses

```
{
  "message": "Invalid JSON body"
}
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Profile updated successfully	None
400	Bad Request	Invalid JSON body	None

Response Schema

Note

This operation does not require authentication

delete_*profiles*{profile_id}

Code samples

```
#  You can also use wget

curl -X DELETE https://PEM-SERVER-IP/pem/api/v16/profiles/{profile_id} \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  X-Auth-Token: string
}

r = requests.delete(https://PEM-SERVER-IP/pem/api/v16/profiles/{profile_id}, headers␣
↪= headers)

print(r.json())
```

```
const headers = {
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v16/profiles/{profile_id},
{
  method: DELETE,

  headers: headers
})
.then(function(res) {
```

(continues on next page)

(continued from previous page)

```
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

```
DELETE /profiles/{profile_id}
```

- Delete a profile*

Parameters

Name	In	Type	Required	Description
profile_id	path	integer	true	none
X-Auth-Token	header	string	true	Token for authorization

Example responses

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Profile deleted successfully	None

Response Schema

> **Note**
>
> This operation does not require authentication

get_*profiles*{profile_id}_assigned

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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/v16/profiles/{profile_id}/assigned,␣
```

(continues on next page)

(continued from previous page)

```
↪headers = headers)

print(r.json())
```

```
const headers = {
  Accept:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v16/profiles/{profile_id}/assigned,
{
  method: GET,

  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

`GET /profiles/{profile_id}/assigned`

- Get servers and agents assigned to a profile*

Parameters

Name	In	Type	Required	Description
profile_id	path	integer	true	none
X-Auth-Token	header	string	true	Token for authorization

Example responses

```
{
  "servers": [
    "string"
  ],
  "agents": [
    "string"
  ]
}
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	Inline

Response Schema

Status Code **200**

Name	Type	Required	Restrictions	Description
» servers	[string]	false	none	none
» agents	[string]	false	none	none

Note

This operation does not require authentication

get_*profiles*{profile_id}_draft

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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/v16/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/v16/profiles/{profile_id}/draft,
{
  method: GET,

  headers: headers
})
```

(continues on next page)

(continued from previous page)

```
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

```
GET /profiles/{profile_id}/draft
```

- Get a draft by profile id*

Returns the draft profile associated with the given published profile id, including full probe and alert configurations. The response does not include parent_id or draft_id fields.

Parameters

Name	In	Type	Required	Description
profile_id	path	integer	true	none
X-Auth-Token	header	string	true	Token for authorization

Example responses


```
{
  "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,
```


(continues on next page)

(continued from previous page)


```
      "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
      ]
    }
  ]
}
```


Draft not found for profile

```
{
  "message": "Not found"
}
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	*ProfileDraft*
404	Not Found	Draft not found for profile	Inline

Response Schema

Status Code **404**

Name	Type	Required	Restrictions	Description
» message	string	false	none	none

Note

This operation does not require authentication

delete_*profiles*{profile_id}_rollback

Code samples

```
#  You can also use wget

curl -X DELETE https://PEM-SERVER-IP/pem/api/v16/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/v16/profiles/{profile_id}/rollback,␣
↪headers = headers)

print(r.json())
```

```
const headers = {
  X-Auth-Token:string
};
```

(continues on next page)

(continued from previous page)

```
fetch(https://PEM-SERVER-IP/pem/api/v16/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
```

- Rollback draft for a profile*

Parameters

Name	In	Type	Required	Description
profile_id	path	integer	true	none
X-Auth-Token	header	string	true	Token for authorization

Example responses

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Draft deleted or no draft available	None

Response Schema

Note

This operation does not require authentication

post_*profiles*{profile_id}_publish

Code samples

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v16/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/v16/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/v16/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`

- Publish a profile*

Publishes the draft for the given profile if present. Returns a message indicating the profile is already published when no draft exists.

Parameters

Name	In	Type	Required	Description
profile_id	path	integer	true	none
X-Auth-Token	header	string	true	Token for authorization

Example responses

```
{
  "message": "Not found"
}
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Profile published successfully or already published	None
404	Not Found	Profile not found	Inline

Response Schema

Status Code **404**

Name	Type	Required	Restrictions	Description
» message	string	false	none	none

Note

This operation does not require authentication

delete_*profiles*{profile_id}_{new_profile_id}

Code samples

```
#  You can also use wget

curl -X DELETE https://PEM-SERVER-IP/pem/api/v16/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/v16/profiles/{profile_id}/{new_
↪profile_id}, headers = headers)

print(r.json())
```

```
const headers = {
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v16/profiles/{profile_id}/{new_profile_id},
{
  method: DELETE,

  headers: headers
```

(continues on next page)

(continued from previous page)

```
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

`DELETE /profiles/{profile_id}/{new_profile_id}`

- Delete a profile and reassign to new profile*

Parameters

Name	In	Type	Required	Description
profile_id	path	integer	true	none
new_profile_id	path	integer	true	none
X-Auth-Token	header	string	true	Token for authorization

Example responses

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Profile deleted and reassigned successfully	None

Response Schema

Note

This operation does not require authentication

Monitored Object Status

get_*agent_status*

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/agent/status/ \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
```

(continues on next page)

(continued from previous page)

```
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v16/agent/status/, headers = headers)

print(r.json())
```

```
const headers = {
  Accept:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v16/agent/status/,
{
  method: GET,

  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

`GET /agent/status/`

- Retrieve the status of all visible pem-agents.*

Returns a list of information and status for the pem-agents visible to the user (represented by the token).

Parameters

Name	In	Type	Required	Description
X-Auth-Token	header	string	true	Token for authentication purpose

Example responses

```
[
  {
    "id": 0,
    "name": "string",
    "group_id": 0,
    "group_name": "string",
    "blackout": true,
    "status": "UP",
    "alerts": {
      "total": 0,
      "acknowledged": 0,
      "high": 0,
      "medium": 0,
```

(continues on next page)

(continued from previous page)


```
      "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
  }
]
```


Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Returns a list of information and status about pem-agents.	Inline

Response Schema

Status Code **200**

Name	Type	Required	Restrictions	Description
anonymous	*Query information*]	false	none	none
» id	integer	false	none	Agent ID
» name	string	false	none	Agent name
» group_id	integer	false	none	Group ID under which agent is visible to the user
» group_name	string	false	none	Group name
» blackout	boolean	false	none	Determines whether an alert blackout for the agent is enabled or disabled.
» status	string	false	none	Agent status can be UP, DOWN, or UNKNOWN
» alerts	object	false	none	List of alerts configured for the agent, including alert status and number of alerts that are acknowledged
»» total	integer	false	none	Number of alerts configured for the agent
»» acknowledged	integer	false	none	Number of acknowledged alerts

continues on next page

Table 103 – continued from previous page

Name	Type	Required	Restrictions	Description
»» high	integer	false	none	Number of alerts with state as 'High'
»» medium	integer	false	none	Number of alerts with state as 'Medium'
»» low	integer	false	none	Number of alerts with state as 'Low'
»» high_acknowledged	integer	false	none	Number of acknowledged alerts with state as 'High'
»» medium_acknowledged	integer	false	none	Number of acknowledged alerts with state as 'Medium'
»» low_acknowledged	integer	false	none	Number of acknowledged alerts with state as 'Low'
» version	string	false	none	Current version of the agent
» processes	integer	false	none	Number of processes running
» threads	integer	false	none	Number of threads for all the running processes
» cpu_utilization	number	false	none	CPU utilization (percentage)
» memory_utilization	number	false	none	Memory utilization (percentage)
» swap_utilization	number	false	none	Swap memory utilization (percentage)
» disk_utilization	number	false	none	Overall disk utilization (percentage)

Enumerated Values

Property	Value
status	UP
status	DOWN
status	UNKNOWN

Note

This operation does not require authentication

get_*agent_status*{agent_id}

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/agent/status/{agent_id} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
```

(continues on next page)

(continued from previous page)

```
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v16/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/v16/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}`

- Retrieve the status of a pem-agent, identified by the agent_id.*

Parameters

Name	In	Type	Required	Description
agent_id	path	integer	true	none
X-Auth-Token	header	string	true	Token of authorization

Example responses

```
{
  "id": 0,
  "name": "string",
  "group_id": 0,
  "group_name": "string",
  "blackout": true,
  "status": "UP",
  "alerts": {
    "total": 0,
    "acknowledged": 0,
    "high": 0,
    "medium": 0,
```

(continues on next page)

(continued from previous page)


```
    "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
}
```


Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	*AgentStatus*
400	Bad Request	Agent not found or visible to the user	None

Response Schema

Note

This operation does not require authentication

get_*server_status*

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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/v16/server/status/, headers = headers)

print(r.json())
```

```
const headers = {
  Accept:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v16/server/status/,
{
  method: GET,

  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

```
GET /server/status/
```

- Retrieve the status of all the visible database servers.*

Returns a list of information for and status of the database servers visible to the user (represented by the token).

Parameters

Name	In	Type	Required	Description
X-Auth-Token	header	string	true	Token for authentication purposes

Example responses

```
[
  {
    "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,
```

(continues on next page)

(continued from previous page)


```
    "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
        }
      ]
    }
  }
]
```


Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Returns a list of information for and status of database servers.	Inline

Response Schema

Status Code **200**

Name	Type	Required	Restrictions	Description
anonymous	*Query information*]	false	none	none
» id	integer	false	none	Server ID
» name	string	false	none	Server name

continues on next page

Table 109 – continued from previous page

Name	Type	Required	Restrictions	Description
» group_id	integer	false	none	Group ID under which database server is visible to the user
» group_name	string	false	none	Group name
» blackout	boolean	false	none	Status of the alert blackout for the database server
» status	string	false	none	Server status can be UP, DOWN, UNKNOWN, or UNMANAGED,
» alerts	object	false	none	List of alerts configured for the database server, including alert status and number of alerts that are acknowledged
»» total	integer	false	none	Number of alerts configured for the database server
»» acknowledged	integer	false	none	Number of acknowledged alerts
»» high	integer	false	none	Number of alerts with state as 'High'
»» medium	integer	false	none	Number of alerts with state as 'Medium'
»» low	integer	false	none	Number of alerts with state as 'Low'
»» high_acknowledged	integer	false	none	Number of acknowledged alerts with state as 'High'
»» medium_acknowledged	integer	false	none	Number of acknowledged alerts with state as 'Medium'
»» low_acknowledged	integer	false	none	Number of acknowledged alerts with state as 'Low'
» version	string	false	none	Current version of the database server
» number_connections	integer	false	none	Number of connections on the database servers
» remote_monitoring	boolean	false	none	Determines whether the database server is being monitored remotely by the pemAgent
» agent_id	integer	false	none	Agent ID that is monitoring the database server
» sessions	object	false	none	All the session information for database servers
»» last_recorded_time	number	false	none	Unix time of the last recorded time for session information
»» sessions	[object]	false	none	List of all the session information for the database server
»»» database_name	string	false	none	Database name
»»» procpid	integer	false	none	Process ID
»»» usename	string	false	none	User name
»»» backend_start	number	false	none	Unix time for the session start time
»»» xact_start	number	false	none	Unix time for the transaction start time

continues on next page

Table 109 – continued from previous page

Name	Type	Required	Restrictions	Description
»»»» query_start	number	false	none	Unix time for the query start time (if any queries are running)
»»»» is_waiting	boolean	false	none	Is the sesion waiting?
»»»» is_idle	boolean	false	none	Is the session idle?
»»»» is_idle_in_transaction	boolean	false	none	Is the session idle in a transaction?
»»»» is_vacuum	boolean	false	none	Is this a VACUUM session?
»»»» is_autovacuum	boolean	false	none	Is this an AUTO VACUUM session?
»»»» client_addr	string	false	none	Host address for the client of the session
»»»» client_port	integer	false	none	Client port for the session
»»»» memory_usage_mb	number	false	none	Memory usage (in MB) of the session (available only when monitored locally)
»»»» swap_usage_mb	number	false	none	Swap memory usage (in MB) of the session (available only when monitored locally)
»»»» cpu_usage	number	false	none	CPU Usage (in percentage) of the session (available only when monitored locally)
»»»» io_read_bytes	number	false	none	I/O Read (in bytes) of the session (available only when monitored locally)
»»»» io_write_bytes	number	false	none	I/O Write (in bytes) of the session (available only when monitored locally)
»»»» state	string	false	none	Current state of the session (idle, active, idle in transcation, and so forth)
»»»» state_change	number	false	none	Unix time of the last state change in the session

Enumerated Values

Property	Value
status	UP
status	DOWN
status	UNKNOWN
status	UNMANAGED

Note

This operation does not require authentication

get_*server_status*{server_id}

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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/v16/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/v16/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}`

- Retrieves the status of a database server, identified by the server_id.*

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	none
X-Auth-Token	header	string	true	Token of authorization

Example responses

```
{
  "id": 0,
```

(continues on next page)

(continued from previous page)


```
  "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
      }
    ]
  }
}
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	*ServerStatus*
400	Bad Request	Server not found or visible to the user	None

Response Schema

> **Note**
>
> This operation does not require authentication

Agent Job

get_*agent_job*{agent_id}

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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/v16/agent/job/{agent_id}, headers =␣
↪headers)

print(r.json())
```

```
const headers = {
  Accept:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v16/agent/job/{agent_id},
{
  method: GET,

  headers: headers
})
.then(function(res) {
```

(continues on next page)

(continued from previous page)

```
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

`GET /agent/job/{agent_id}`

Gets `Jobs` for agent level by agent_id.

Parameters

Name	In	Type	Required	Description
agent_id	path	integer	true	Agent ID
X-Auth-Token	header	string	true	Token of authorization

Example responses

```
[
  {
    "jobname": "string",
    "jobenabled": true,
    "jobdesc": "string"
  }
]
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	Inline

Response Schema

Status Code **200**

Name	Type	Required	Restrictions	Description
anonymous	*Query information*]	false	none	none
» jobname	string	false	none	Name of the job.
» jobenabled	boolean	false	none	Determines whether the job is enabled or disabled.
» jobdesc	string	false	none	Description of the job.

Note

This operation does not require authentication

post_*agent_job*{agent_id}

Code samples

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v16/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/v16/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": "Monday",
      "jscmonthdays": 1,
      "jscmonths": "January",
      "jschours": 0,
      "jscminutes": 0,
      "jscexceptions": [
        {
          "jexdate": "2023-10-19",
          "jextime": "09:39:00"
        }
      ],
      "jsctimezone": "Asia/Calcutta"
    }
  ],
  "notify": "DEFAULT",
```

(continues on next page)

(continued from previous page)

```
  "email_group_id": "string"
};
const headers = {
  Content-Type:application/json,
  Accept:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v16/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}`

Creates `Job` for agent level.

Body parameter

```
{
  "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": "Monday",
      "jscmonthdays": 1,
      "jscmonths": "January",
      "jschours": 0,
      "jscminutes": 0,
      "jscexceptions": [
        {
          "jexdate": "2023-10-19",
          "jextime": "09:39:00"
        }
      ],
      "jsctimezone": "Asia/Calcutta"
    }
  ],
  "notify": "DEFAULT",
```

(continues on next page)

(continued from previous page)

```
  "email_group_id": "string"
}
```

Parameters

Name	In	Type	Required	Description
agent_id	path	integer	true	Agent ID
X-Auth-Token	header	string	true	Token of authorization
body	body	*AgentJobBody-Post*	true	none

Example responses

```
{
  "id": 0
}
```

Responses

Status	Meaning	Description	Schema
201	Created	Created Successfully	Inline

Response Schema

Status Code **201**

Name	Type	Required	Restrictions	Description
» id	integer	false	none	Id of the job

Note

This operation does not require authentication

get_*agent_job*{agent_id}_{job_id}

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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/v16/agent/job/{agent_id}/{job_id},␣
↪headers = headers)

print(r.json())
```

```
const headers = {
  Accept:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v16/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}`

Gets `Jobs` for agent level by agent_id and job_id.

Parameters

Name	In	Type	Required	Description
agent_id	path	integer	true	Agent ID
job_id	path	integer	true	Job ID
X-Auth-Token	header	string	true	Token of authorization

Example responses

```
{
  "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",
```

(continues on next page)

(continued from previous page)


```
  "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": [
        true
      ],
      "jscmonthdays": [
        true
      ],
      "jscmonths": [
        true
      ],
      "jschours": [
        true
      ],
      "jscminutes": [
        true
      ],
      "jscexceptions": [
        null
      ],
      "jsctimezone": "string"
    }
  ]
}
```


Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	*AgentJobBodyGet*

Note

This operation does not require authentication

put_*agent_job*{agent_id}_{job_id}

Code samples

```
#  You can also use wget

curl -X PUT https://PEM-SERVER-IP/pem/api/v16/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
}

r = requests.put(https://PEM-SERVER-IP/pem/api/v16/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"
      }
    ],
    "changed": [
      {
        "jstid": 0,
        "jstname": "string",
        "jstdesc": "string",
        "jstenabled": true,
        "jstkind": true,
        "jstcode": "string",
        "server_id": 0,
        "database_name": "string",
        "jstonerror": "f",
        "jstnextrun": "string"
      }
    ]
  },
```

(continues on next page)

(continued from previous page)

```
  "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": "Monday",
        "jscmonthdays": 1,
        "jscmonths": "January",
        "jschours": 0,
        "jscminutes": 0,
        "jscexceptions": [
          {
            "jexdate": "2023-10-19",
            "jextime": "09:39:00"
          }
        ],
        "jsctimezone": "Asia/Calcutta"
      }
    ],
    "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": "Monday",
        "jscmonthdays": 1,
        "jscmonths": "January",
        "jschours": 0,
        "jscminutes": 0,
        "jscexceptions": [
          {
            "jexdate": "2023-10-19",
            "jextime": "09:39:00"
          }
        ],
        "jsctimezone": "Asia/Calcutta"
      }
    ]
  },
  "notify": "DEFAULT",
  "email_group_id": "string"
};
const headers = {
  Content-Type:application/json,
  X-Auth-Token:string
};
```

(continues on next page)

(continued from previous page)

```
fetch(https://PEM-SERVER-IP/pem/api/v16/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}`

Updates `Job` for agent level.

Body parameter

```
{
  "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"
      }
    ],
    "changed": [
      {
        "jstid": 0,
        "jstname": "string",
        "jstdesc": "string",
        "jstenabled": true,
        "jstkind": true,
        "jstcode": "string",
        "server_id": 0,
        "database_name": "string",
        "jstonerror": "f",
        "jstnextrun": "string"
      }
    ]
  },
  "jschedules": {
    "added": [
```

(continues on next page)

(continued from previous page)


```
      {
        "jscname": "string",
        "jscdesc": "string",
        "jscenabled": true,
        "jscstart": "2023-10-09 22:43 +05:30",
        "jscend": "2023-10-11 22:43 +05:30",
        "jscweekdays": "Monday",
        "jscmonthdays": 1,
        "jscmonths": "January",
        "jschours": 0,
        "jscminutes": 0,
        "jscexceptions": [
          {
            "jexdate": "2023-10-19",
            "jextime": "09:39:00"
          }
        ],
        "jsctimezone": "Asia/Calcutta"
      }
    ],
    "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": "Monday",
        "jscmonthdays": 1,
        "jscmonths": "January",
        "jschours": 0,
        "jscminutes": 0,
        "jscexceptions": [
          {
            "jexdate": "2023-10-19",
            "jextime": "09:39:00"
          }
        ],
        "jsctimezone": "Asia/Calcutta"
      }
    ]
  },
  "notify": "DEFAULT",
  "email_group_id": "string"
}
```


Parameters

Name	In	Type	Required	Description
agent_id	path	integer	true	Agent ID
job_id	path	integer	true	Job ID
X-Auth-Token	header	string	true	Token of authorization
body	body	*AgentJobBody-Put*	true	none

Example responses

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Status 200	None

Response Schema

> **Note**
>
> This operation does not require authentication

delete_*agent_job*{agent_id}_{job_id}

Code samples

```
#  You can also use wget

curl -X DELETE https://PEM-SERVER-IP/pem/api/v16/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/v16/agent/job/{agent_id}/{job_id},␣
↪headers = headers)

print(r.json())
```

```
const headers = {
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v16/agent/job/{agent_id}/{job_id},
{
  method: DELETE,
```

(continues on next page)

(continued from previous page)

```
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

`DELETE /agent/job/{agent_id}/{job_id}`

Deletes the `Job` using agent_id and job_id.

Parameters

Name	In	Type	Required	Description
agent_id	path	integer	true	Agent ID
job_id	path	integer	true	Job ID
X-Auth-Token	header	string	true	Token of authorization

Example responses

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful deletion	None

Response Schema

Note

This operation does not require authentication

Alerts History

get_*alert_history*

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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/v16/alert/history/, headers = headers)

print(r.json())
```

```
const headers = {
  Accept:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v16/alert/history/,
{
  method: GET,

  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

`GET /alert/history/`

Gets the state change history of all the alerts.

Parameters

Name	In	Type	Required	Description
X-Auth-Token	header	string	true	Token of authorization

Example responses

```
[
  {
    "alert_id": 0,
    "state": "High",
    "value": "string",
    "actual_value": 0
  }
]
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	Inline

Response Schema

Status Code **200**

Name	Type	Required	Restrictions	Description
anonymous	*Query information*]	false	none	none
» alert_id	integer	false	none	Alert ID
» state	string	false	none	State
» value	string	false	none	User-presentable value
» actual_value	number	false	none	Actual value at the time of state change

Enumerated Values

Property	Value
state	High
state	Medium
state	Low

Note

This operation does not require authentication

get_*alert_history_agent*{agent_id}

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/alert/history/agent/{agent_id} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}
```

(continues on next page)

(continued from previous page)

```
r = requests.get(https://PEM-SERVER-IP/pem/api/v16/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/v16/alert/history/agent/{agent_id},
{
  method: GET,

  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

`GET /alert/history/agent/{agent_id}`

Gets the state change history of all the alerts for the agent.

Parameters

Name	In	Type	Required	Description
agent_id	path	integer	true	Valid Agent ID
X-Auth-Token	header	string	true	Token of authorization

Example responses

```
[
  {
    "alert_id": 0,
    "state": "High",
    "value": "string",
    "actual_value": 0
  }
]
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	Inline

Response Schema

Status Code **200**

Name	Type	Required	Restrictions	Description
anonymous	*Query information*]	false	none	none
» alert_id	integer	false	none	Alert ID
» state	string	false	none	State
» value	string	false	none	User-presentable value
» actual_value	number	false	none	Actual value at the time of state change

Enumerated Values

Property	Value
state	High
state	Medium
state	Low

Note

This operation does not require authentication

get_*alert_history_server*{server_id}

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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/v16/alert/history/server/{server_id},␣
↪headers = headers)
```

(continues on next page)

(continued from previous page)

```
print(r.json())
```

```
const headers = {
  Accept:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v16/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}`

Gets the state change history of all the alerts for the database server.

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Valid Server ID
X-Auth-Token	header	string	true	Token of authorization

Example responses

```
[
  {
    "alert_id": 0,
    "state": "High",
    "value": "string",
    "actual_value": 0
  }
]
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	Inline

Response Schema

Status Code **200**

Name	Type	Required	Restrictions	Description
anonymous	*Query information*]	false	none	none
» alert_id	integer	false	none	Alert ID
» state	string	false	none	State
» value	string	false	none	User-presentable value
» actual_value	number	false	none	Actual value at the time of state change

Enumerated Values

Property	Value
state	High
state	Medium
state	Low

Note

This operation does not require authentication

get_*alert_history_server*{server_id}*database*{database_name}

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v16/alert/history/server/{server_id}/
↪database/{database_name}, headers = headers)

print(r.json())
```

```
const headers = {
  Accept:application/json,
```

(continues on next page)

(continued from previous page)

```
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v16/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}`

Gets the state change history of all the alerts for the database of the database server.

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Valid Server ID
database_name	path	string	true	Database name
X-Auth-Token	header	string	true	Token of authorization

Example responses

```
[
  {
    "alert_id": 0,
    "state": "High",
    "value": "string",
    "actual_value": 0
  }
]
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	Inline

Response Schema

Status Code **200**

Name	Type	Required	Restrictions	Description
anonymous	*Query information*]	false	none	none
» alert_id	integer	false	none	Alert ID
» state	string	false	none	State
» value	string	false	none	User-presentable value
» actual_value	number	false	none	Actual value at the time of state change

Enumerated Values

Property	Value
state	High
state	Medium
state	Low

Note

This operation does not require authentication

Alerts Status

get_*alert_status*

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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/v16/alert/status/, headers = headers)

print(r.json())
```

```
const headers = {
  Accept:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v16/alert/status/,
{
  method: GET,

  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

`GET /alert/status/`

Gets the current status of all the alerts.

Parameters

Name	In	Type	Required	Description
X-Auth-Token	header	string	true	Token of authorization

Example responses

```
[
  {
    "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"
```

(continues on next page)

(continued from previous page)

```
        ]
      ]
    }
]
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	Inline

Response Schema

Status Code **200**

Name	Type	Required	Restrictions	Description
anonymous	*Query information*]	false	none	none
» alert_since	string	false	none	Unix time when alert state last changed
» alert_id	integer	false	none	Alert internal identification number
» alert_name	string	false	none	Name of the alert
» alert_state	string	false	none	Current state of the alert. A 'null' value suggests alert threshold values have not currently been violated for the alert.
» value	string	false	none	Current value of the alert
» object_description	string	false	none	Monitoring object description/name
» server_id	integer	false	none	null or ID of the server
» agent_id	integer	false	none	null or ID of the agent
» database	string	false	none	null or name of the database
» schema	string	false	none	null or name of the schema/namespace
» package	string	false	none	null or name of the EDB package
» object	string	false	none	null or name of any other type of object
» alert_target_level	string	false	none	Alert target level
» last_processed	string	false	none	String reprenting Unix time when the alert was processed

continues on next page

Table 143 – continued from previous page

Name	Type	Required	Restrictions	Description
» info	string	false	none	Some information at the time when the alert was generated
» info_cols	[string]	false	none	Array of column names for the information
» info_vals	[array]	false	none	Detailed information associated with the alert in the form of a table (an array of an array)

Enumerated Values

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
alert_target_level	Index
alert_target_level	Sequence
alert_target_level	Function
alert_target_level	View

Note

This operation does not require authentication

get_*alert_status*{since}

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/alert/status/{since} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
  X-Auth-Token: string
}
```

(continues on next page)

(continued from previous page)

```
r = requests.get(https://PEM-SERVER-IP/pem/api/v16/alert/status/{since}, headers =␣
↪headers)

print(r.json())
```

```
const headers = {
  Accept:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v16/alert/status/{since},
{
  method: GET,

  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

`GET /alert/status/{since}`

Gets the list of alert state changes after the given time.

Parameters

Name	In	Type	Required	Description
since	path	number	true	Unix timeline
X-Auth-Token	header	string	true	Token of authorization

Example responses

```
{
  "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",
```

(continues on next page)

(continued from previous page)

```
  "info": "string",
  "info_cols": [
    "string"
  ],
  "info_vals": [
    [
      "string"
    ]
  ]
}
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	*AlertStatus*

Note

This operation does not require authentication

Alerts

get_alert_config_global

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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/v16/alert/config/global/, headers =␣
↪headers)

print(r.json())
```

```
const headers = {
  Accept:application/json,
  X-Auth-Token:string
```

(continues on next page)

(continued from previous page)

```
};

fetch(https://PEM-SERVER-IP/pem/api/v16/alert/config/global/,
{
  method: GET,

  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

`GET /alert/config/global/`

Gets `Alerts` for global level.

Parameters

Name	In	Type	Required	Description
X-Auth-Token	header	string	true	Token of authorization

Example responses

```
[
  {
    "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,
```

(continues on next page)

(continued from previous page)


```
    "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
    ]
  }
]
```


Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	Inline

Response Schema

Status Code **200**

Name	Type	Required	Restrictions	Description
anonymous	*Query information*]	false	none	none
» id	integer	false	none	Alert ID. NOTE: This property is not applicable for a POST request.
» alert_name	string	false	none	Name of the alert. NOTE: This property is a mandatory parameter for a POST request.
» alert_template	integer	false	none	ID of the alert template. This property must be a positive integer value. NOTE: It is a mandatory parameter for a POST request.
» description	string	false	none	none
» enabled	boolean	false	none	Determines whether this alert is enabled/disabled. NOTE: This property is a mandatory parameter for a POST request.
» history_retention	integer	false	none	Specifies how many days to keep the history of this alert. Possible values are from 1 to 99999. NOTE: This property is a mandatory parameter for a POST request.
» frequency_min	integer	false	none	Defines in minutes how frequently the alert must be checked on a PEM database server. Possible values are from 1 to 65534. NOTE: This property is a mandatory parameter for a POST request.
» auto_created	boolean	false	none	Specifies whether the alert is auto-created.
» template_id	string	false	none	Defines the alert template.
» params	*Query information*]	false	none	List of parameter/value pairs to be used by the alert template for checking alerts. NOTE: The alert template requires the all parameter to be defined.
»» paramname	string	false	none	Name of the parameter. NOTE: This property must present on the alert template parameter list.Example:

continues on next page

Table 149 – continued from previous page

Name	Type	Required	Restrictions	Description
»» paramvalue	string	false	none	Value for the given parameter
» param_names	[string]	false	none	Defines the name of parameters.
» param_types	string	false	none	Defines the type of parameters. Example: INTEGER/STRING..
» params_units	[string]	false	none	Defines the unit used for parameters.Example: 'DAYS','True or False','Hours','Minutes' etc
» threshold_unit	string	false	none	Units used to calculate the threshold, such as MB/kB/seconds/days/... This value is only used for the GUI.
» operator	string	false	none	Defines the operator for checking threshold values. Valid strings are ">" and "<". NOTE: This property is a mandatory parameter for a POST request.
» low_threshold_value	number	false	none	Threshold value for triggering the low alert. This property can be any valid integer/float value. NOTE: It is a mandatory parameter for the POST request.
» medium_threshold_value	number	false	none	Threshold value for triggering the medium alert. This property can be any valid integer/float value. NOTE: It is a mandatory parameter for the POST request.
» high_threshold_value	number	false	none	Threshold value for triggering the high alert. It can be any valid integer/float value. NOTE: It is a mandatory parameter for the POST request.
» send_email	boolean	false	none	Determines whether to send an email when the state of the alert changes.

continues on next page

Table 149 – continued from previous page

Name	Type	Required	Restrictions	Description
» all_alert_enable	boolean	false	none	Determines whether to send an email notification to the group specified by the parameter 'email_group_id' when the state of the alert changes.
» email_group_id	integer	false	none	Defines the email group to which email notification are sent when the state of the alert changes.
» low_alert_enable	boolean	false	none	Determines whether to send an email notification to the group specified by the parameter 'low_email_group_id' when the state of the alert is set to LOW.
» low_email_group_id	integer	false	none	Defines the email group to which email notification are sent when the state of the alert changes to LOW.
» med_alert_enable	boolean	false	none	Determines whether to send an email notification to the group specified by the parameter 'medium_email_group_id' when the state of the alert is set to MEDIUM.
» med_email_group_id	integer	false	none	Defines the email group to which email notification are sent when state of the alert changes to MEDIUM.
» high_alert_enable	boolean	false	none	Determines whether to send an email notification to the group specified by the parameter 'high_email_group_id' when the state of the alert is set to HIGH.
» high_email_group_id	integer	false	none	Defines the email group to which email notification are sent when the state of the alert changes to HIGH.
» send_trap	boolean	false	none	Determines whether to send SNMP trap notifications when the state of the alert changes.

continues on next page

Table 149 – continued from previous page

Name	Type	Required	Restrictions	Description
» snmp_trap_version	integer	false	none	Defines the SNMP trap version. Supported SNMP trap versions are 1, 2 or 3
» low_send_trap	boolean	false	none	Determines whether to send SNMP trap notifications when the state of the alert changes to LOW.
» med_send_trap	boolean	false	none	Determines whether to send SNMP trap notification when the state of the alert changes to MEDIUM.
» high_send_trap	boolean	false	none	Determines whether to send SNMP trap notification when the state of the alert changes to HIGH.
» cleared_alert_enable	boolean	false	none	Determines whether to mute the notification for 'clear alert'.
» execute_script	boolean	false	none	Determines whether to execute the provided script if the state of the alert changes to LOW/MEDIUM/HIGH.
» execute_script_on_clear	boolean	false	none	Determines whether to execute the provided script if the state of the alert is cleared.
» execute_script_on_pem_server	boolean	false	none	Determines whether to execute the provided script on the PEM host or the host where the agent is monitoring the object.
» script_code	string	false	none	The shell or batch script to be executed. Please read the documentation for detailed information.
» submit_to_nagios	boolean	false	none	Determines whether to send notification to the nagios when the state of the alert changes.
» override_default_config	boolean	false	none	Determines whether to override the default webhook configuration.
» send_notification	boolean	false	none	Determines whether to enable the webhook send notification.
» low_webhook_ids	[integer]	false	none	Webhook IDs for which low alerts are configured

continues on next page

Table 149 – continued from previous page

Name	Type	Required	Restrictions	Description
» med_webhook_ids	[integer]	false	none	Webhook IDs for which medium alerts are configured
» high_webhook_ids	[integer]	false	none	Webhook IDs for which high alerts are configured
» cleared_webhook_ids	[integer]	false	none	Webhook IDs for which cleared alerts are configured

Enumerated Values

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%)

Note

This operation does not require authentication

post_*alert_config_global*

Code samples

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v16/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/v16/alert/config/global/, headers =␣
↪headers)

print(r.json())
```

```
const inputBody = {
  "alert_name": "string",
```

(continues on next page)

(continued from previous page)

```
  "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,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v16/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/`

Creates `Alert` for global level.

Body parameter


```
{
  "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": []
}
```


Parameters

Name	In	Type	Required	Description
X-Auth-Token	header	string	true	Token of authorization
body	body	*AlertPayload*	true	none

Example responses

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Status 200	None

Response Schema

> **Note**
>
> This operation does not require authentication

get_*alert_config_global*{alert_id}

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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/v16/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/v16/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}`

Gets `Alerts` for global level by alert_id.

Parameters

Name	In	Type	Required	Description
alert_id	path	integer	true	Alert ID
X-Auth-Token	header	string	true	Token of authorization

Example responses


```
{
  "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,
```


(continues on next page)

(continued from previous page)

```
  "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
  ]
}
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	*About alert threads*

Note

This operation does not require authentication

put_*alert_config_global*{alert_id}

Code samples

```
#  You can also use wget

curl -X PUT https://PEM-SERVER-IP/pem/api/v16/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/v16/alert/config/global/{alert_id},␣
↪headers = headers)
```

(continues on next page)

(continued from previous page)

```
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,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v16/alert/config/global/{alert_id},
{
  method: PUT,
  body: inputBody,
  headers: headers
})
```

(continues on next page)

(continued from previous page)

```
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

`PUT /alert/config/global/{alert_id}`

Updates `Alert` for global level.

Body parameter

```
{
  "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": []
}
```

Parameters

Name	In	Type	Required	Description
alert_id	path	integer	true	Alert ID
X-Auth-Token	header	string	true	Token of authorization
body	body	*AlertPayload*	true	none

Example responses

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Status 200	None

Response Schema

Note

This operation does not require authentication

delete_*alert_config_global*{alert_id}

Code samples

```
#  You can also use wget

curl -X DELETE https://PEM-SERVER-IP/pem/api/v16/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/v16/alert/config/global/{alert_id},␣
↪headers = headers)

print(r.json())
```

```
const headers = {
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v16/alert/config/global/{alert_id},
{
  method: DELETE,
```

(continues on next page)

(continued from previous page)

```
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

```
DELETE /alert/config/global/{alert_id}
```

Deletes the `Alert` object.

Parameters

Name	In	Type	Required	Description
alert_id	path	integer	true	Alert ID
X-Auth-Token	header	string	true	Token of authorization

Example responses

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful deletion	None

Response Schema

Note

This operation does not require authentication

get_*alert_config_agent*{agent_id}

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/alert/config/agent/{agent_id} \
  -H Accept: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Accept: application/json,
```

(continues on next page)

(continued from previous page)

```
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v16/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/v16/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}`

Gets `Alerts` for agent level by agent_id.

Parameters

Name	In	Type	Required	Description
agent_id	path	integer	true	Agent ID
X-Auth-Token	header	string	true	Token of authorization

Example responses

```
[
  {
    "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": [
```

(continues on next page)

(continued from previous page)


```
      "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
    ]
  }
]
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	Inline

Response Schema

Status Code **200**

Name	Type	Required	Restrictions	Description
anonymous	*Query information*]	false	none	none
» id	integer	false	none	Alert ID. NOTE: This property is not applicable for a POST request.
» alert_name	string	false	none	Name of the alert. NOTE: This property is a mandatory parameter for a POST request.
» alert_template	integer	false	none	ID of the alert template. This property must be a positive integer value. NOTE: It is a mandatory parameter for a POST request.
» description	string	false	none	none
» enabled	boolean	false	none	Determines whether this alert is enabled/disabled. NOTE: This property is a mandatory parameter for a POST request.
» history_retention	integer	false	none	Specifies how many days to keep the history of this alert. Possible values are from 1 to 99999. NOTE: This property is a mandatory parameter for a POST request.
» frequency_min	integer	false	none	Defines in minutes how frequently the alert must be checked on a PEM database server. Possible values are from 1 to 65534. NOTE: This property is a mandatory parameter for a POST request.
» auto_created	boolean	false	none	Specifies whether the alert is auto-created.

continues on next page

Table 161 – continued from previous page

Name	Type	Required	Restrictions	Description
» template_id	string	false	none	Defines the alert template.
» params	*Query information*]	false	none	List of parameter/value pairs to be used by the alert template for checking alerts. NOTE: The alert template requires the all parameter to be defined.
»» paramname	string	false	none	Name of the parameter. NOTE: This property must present on the alert template parameter list.Example:
»» paramvalue	string	false	none	Value for the given parameter
» param_names	[string]	false	none	Defines the name of parameters.
» param_types	string	false	none	Defines the type of parameters. Example: INTEGER/STRING..
» params_units	[string]	false	none	Defines the unit used for parameters.Example: 'DAYS','True or False','Hours','Minutes' etc
» threshold_unit	string	false	none	Units used to calculate the threshold, such as MB/kB/seconds/days/... This value is only used for the GUI.
» operator	string	false	none	Defines the operator for checking threshold values. Valid strings are ">" and "<". NOTE: This property is a mandatory parameter for a POST request.
» low_threshold_value	number	false	none	Threshold value for triggering the low alert. This property can be any valid integer/float value. NOTE: It is a mandatory parameter for the POST request.
» medium_threshold_value	number	false	none	Threshold value for triggering the medium alert. This property can be any valid integer/float value. NOTE: It is a mandatory parameter for the POST request.

continues on next page

Table 161 – continued from previous page

Name	Type	Required	Restrictions	Description
» high_threshold_value	number	false	none	Threshold value for triggering the high alert. It can be any valid integer/float value. NOTE: It is a mandatory parameter for the POST request.
» send_email	boolean	false	none	Determines whether to send an email when the state of the alert changes.
» all_alert_enable	boolean	false	none	Determines whether to send an email notification to the group specified by the parameter 'email_group_id' when the state of the alert changes.
» email_group_id	integer	false	none	Defines the email group to which email notification are sent when the state of the alert changes.
» low_alert_enable	boolean	false	none	Determines whether to send an email notification to the group specified by the parameter 'low_email_group_id' when the state of the alert is set to LOW.
» low_email_group_id	integer	false	none	Defines the email group to which email notification are sent when the state of the alert changes to LOW.
» med_alert_enable	boolean	false	none	Determines whether to send an email notification to the group specified by the parameter 'medium_email_group_id' when the state of the alert is set to MEDIUM.
» med_email_group_id	integer	false	none	Defines the email group to which email notification are sent when state of the alert changes to MEDIUM.
» high_alert_enable	boolean	false	none	Determines whether to send an email notification to the group specified by the parameter 'high_email_group_id' when the state of the alert is set to HIGH.

continues on next page

Table 161 – continued from previous page

Name	Type	Required	Restrictions	Description
» high_email_group_id	integer	false	none	Defines the email group to which email notification are sent when the state of the alert changes to HIGH.
» send_trap	boolean	false	none	Determines whether to send SNMP trap notifications when the state of the alert changes.
» snmp_trap_version	integer	false	none	Defines the SNMP trap version. Supported SNMP trap versions are 1, 2 or 3
» low_send_trap	boolean	false	none	Determines whether to send SNMP trap notifications when the state of the alert changes to LOW.
» med_send_trap	boolean	false	none	Determines whether to send SNMP trap notification when the state of the alert changes to MEDIUM.
» high_send_trap	boolean	false	none	Determines whether to send SNMP trap notification when the state of the alert changes to HIGH.
» cleared_alert_enable	boolean	false	none	Determines whether to mute the notification for 'clear alert'.
» execute_script	boolean	false	none	Determines whether to execute the provided script if the state of the alert changes to LOW/MEDIUM/HIGH.
» execute_script_on_clear	boolean	false	none	Determines whether to execute the provided script if the state of the alert is cleared.
» execute_script_on_pem_server	boolean	false	none	Determines whether to execute the provided script on the PEM host or the host where the agent is monitoring the object.
» script_code	string	false	none	The shell or batch script to be executed. Please read the documentation for detailed information.
» submit_to_nagios	boolean	false	none	Determines whether to send notification to the nagios when the state of the alert changes.

continues on next page

Table 161 – continued from previous page

Name	Type	Required	Restrictions	Description
» override_default_config	boolean	false	none	Determines whether to override the default webhook configuration.
» send_notification	boolean	false	none	Determines whether to enable the webhook send notification.
» low_webhook_ids	[integer]	false	none	Webhook IDs for which low alerts are configured
» med_webhook_ids	[integer]	false	none	Webhook IDs for which medium alerts are configured
» high_webhook_ids	[integer]	false	none	Webhook IDs for which high alerts are configured
» cleared_webhook_ids	[integer]	false	none	Webhook IDs for which cleared alerts are configured

Enumerated Values

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%)

Note

This operation does not require authentication

post_*alert_config_agent*{agent_id}

Code samples

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v16/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
}
```

(continues on next page)

(continued from previous page)

```
r = requests.post(https://PEM-SERVER-IP/pem/api/v16/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,
  "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/v16/alert/config/agent/{agent_id},
{
  method: POST,
```

(continues on next page)

(continued from previous page)

```
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

`POST /alert/config/agent/{agent_id}`

Creates `Alert` for agent level.

Body parameter

```
{
  "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": []
}
```

Parameters

Name	In	Type	Required	Description
agent_id	path	integer	true	Agent ID
X-Auth-Token	header	string	true	Token of authorization
body	body	*AlertPayload*	true	none

Example responses

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Status 200	None

Response Schema

> **Note**
>
> This operation does not require authentication

get_*alert_config_agent*{agent_id}_{alert_id}

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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/v16/alert/config/agent/{agent_id}/
↪{alert_id}, headers = headers)

print(r.json())
```

```
const headers = {
  Accept:application/json,
  X-Auth-Token:string
};
```

(continues on next page)

(continued from previous page)

```
fetch(https://PEM-SERVER-IP/pem/api/v16/alert/config/agent/{agent_id}/{alert_id},
{
  method: GET,

  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

`GET /alert/config/agent/{agent_id}/{alert_id}`

Gets `Alerts` for agent level by agent_id and alert_id.

Parameters

Name	In	Type	Required	Description
agent_id	path	integer	true	Agent ID
alert_id	path	integer	true	Alert ID
X-Auth-Token	header	string	true	Token of authorization

Example responses


```
{
  "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,
```


(continues on next page)

(continued from previous page)


```
  "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
  ]
}
```


Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	*About alert threads*

Note

This operation does not require authentication

put_*alert_config_agent*{agent_id}_{alert_id}

Code samples

```
#  You can also use wget

curl -X PUT https://PEM-SERVER-IP/pem/api/v16/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/v16/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,
  "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,
```

(continues on next page)

(continued from previous page)

```
  "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/v16/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}`

Updates `Alert` for agent level.

Body parameter

```
{
  "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,
```

(continues on next page)

(continued from previous page)

```
  "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": []
}
```

Parameters

Name	In	Type	Required	Description
agent_id	path	integer	true	Agent ID
alert_id	path	integer	true	Alert ID
X-Auth-Token	header	string	true	Token of authorization
body	body	*AlertPayload*	true	none

Example responses

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Status 200	None

Response Schema

> **Note**
>
> This operation does not require authentication

delete_*alert_config_agent*{agent_id}_{alert_id}

Code samples

```
#  You can also use wget

curl -X DELETE https://PEM-SERVER-IP/pem/api/v16/alert/config/agent/{agent_id}/{alert_
```

(continues on next page)

(continued from previous page)

```
↪id} \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  X-Auth-Token: string
}

r = requests.delete(https://PEM-SERVER-IP/pem/api/v16/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/v16/alert/config/agent/{agent_id}/{alert_id},
{
  method: DELETE,

  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

`DELETE /alert/config/agent/{agent_id}/{alert_id}`

Deletes the `Alert` object.

Parameters

Name	In	Type	Required	Description
agent_id	path	integer	true	Agent ID
alert_id	path	integer	true	Alert ID
X-Auth-Token	header	string	true	Token of authorization

Example responses

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful deletion	None

Response Schema

> **Note**
>
> This operation does not require authentication

get_*alert_config_server*{server_id}

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/alert/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/v16/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/v16/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}`

Gets `Alerts` for server level by server_id.

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
X-Auth-Token	header	string	true	Token of authorization

Example responses


```
[
  {
    "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,
```


(continues on next page)

(continued from previous page)


```
    "override_default_config": true,
    "send_notification": true,
    "low_webhook_ids": [
      0
    ],
    "med_webhook_ids": [
      0
    ],
    "high_webhook_ids": [
      0
    ],
    "cleared_webhook_ids": [
      0
    ]
  }
]
```


Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	Inline

Response Schema

Status Code **200**

Name	Type	Required	Restrictions	Description
anonymous	*Query information*]	false	none	none
» id	integer	false	none	Alert ID. NOTE: This property is not applicable for a POST request.
» alert_name	string	false	none	Name of the alert. NOTE: This property is a mandatory parameter for a POST request.
» alert_template	integer	false	none	ID of the alert template. This property must be a positive integer value. NOTE: It is a mandatory parameter for a POST request.
» description	string	false	none	none

continues on next page

Table 173 – continued from previous page

Name	Type	Required	Restrictions	Description
» enabled	boolean	false	none	Determines whether this alert is enabled/disabled. NOTE: This property is a mandatory parameter for a POST request.
» history_retention	integer	false	none	Specifies how many days to keep the history of this alert. Possible values are from 1 to 99999. NOTE: This property is a mandatory parameter for a POST request.
» frequency_min	integer	false	none	Defines in minutes how frequently the alert must be checked on a PEM database server. Possible values are from 1 to 65534. NOTE: This property is a mandatory parameter for a POST request.
» auto_created	boolean	false	none	Specifies whether the alert is auto-created.
» template_id	string	false	none	Defines the alert template.
» params	*Query information*]	false	none	List of parameter/value pairs to be used by the alert template for checking alerts. NOTE: The alert template requires the all parameter to be defined.
»» paramname	string	false	none	Name of the parameter. NOTE: This property must present on the alert template parameter list.Example:
»» paramvalue	string	false	none	Value for the given parameter
» param_names	[string]	false	none	Defines the name of parameters.
» param_types	string	false	none	Defines the type of parameters. Example: INTEGER/STRING..
» params_units	[string]	false	none	Defines the unit used for parameters.Example: 'DAYS','True or False','Hours','Minutes' etc

continues on next page

Table 173 – continued from previous page

Name	Type	Required	Restrictions	Description
» threshold_unit	string	false	none	Units used to calculate the threshold, such as MB/kB/seconds/days/... This value is only used for the GUI.
» operator	string	false	none	Defines the operator for checking threshold values. Valid strings are ">" and "<". NOTE: This property is a mandatory parameter for a POST request.
» low_threshold_value	number	false	none	Threshold value for triggering the low alert. This property can be any valid integer/float value. NOTE: It is a mandatory parameter for the POST request.
» medium_threshold_value	number	false	none	Threshold value for triggering the medium alert. This property can be any valid integer/float value. NOTE: It is a mandatory parameter for the POST request.
» high_threshold_value	number	false	none	Threshold value for triggering the high alert. It can be any valid integer/float value. NOTE: It is a mandatory parameter for the POST request.
» send_email	boolean	false	none	Determines whether to send an email when the state of the alert changes.
» all_alert_enable	boolean	false	none	Determines whether to send an email notification to the group specified by the parameter 'email_group_id' when the state of the alert changes.
» email_group_id	integer	false	none	Defines the email group to which email notification are sent when the state of the alert changes.

continues on next page

Table 173 – continued from previous page

Name	Type	Required	Restrictions	Description
» low_alert_enable	boolean	false	none	Determines whether to send an email notification to the group specified by the parameter 'low_email_group_id' when the state of the alert is set to LOW.
» low_email_group_id	integer	false	none	Defines the email group to which email notification are sent when the state of the alert changes to LOW.
» med_alert_enable	boolean	false	none	Determines whether to send an email notification to the group specified by the parameter 'medium_email_group_id' when the state of the alert is set to MEDIUM.
» med_email_group_id	integer	false	none	Defines the email group to which email notification are sent when state of the alert changes to MEDIUM.
» high_alert_enable	boolean	false	none	Determines whether to send an email notification to the group specified by the parameter 'high_email_group_id' when the state of the alert is set to HIGH.
» high_email_group_id	integer	false	none	Defines the email group to which email notification are sent when the state of the alert changes to HIGH.
» send_trap	boolean	false	none	Determines whether to send SNMP trap notifications when the state of the alert changes.
» snmp_trap_version	integer	false	none	Defines the SNMP trap version. Supported SNMP trap versions are 1, 2 or 3
» low_send_trap	boolean	false	none	Determines whether to send SNMP trap notifications when the state of the alert changes to LOW.
» med_send_trap	boolean	false	none	Determines whether to send SNMP trap notification when the state of the alert changes to MEDIUM.

continues on next page

Table 173 – continued from previous page

Name	Type	Required	Restrictions	Description
» high_send_trap	boolean	false	none	Determines whether to send SNMP trap notification when the state of the alert changes to HIGH.
» cleared_alert_enable	boolean	false	none	Determines whether to mute the notification for 'clear alert'.
» execute_script	boolean	false	none	Determines whether to execute the provided script if the state of the alert changes to LOW/MEDIUM/HIGH.
» execute_script_on_clear	boolean	false	none	Determines whether to execute the provided script if the state of the alert is cleared.
» execute_script_on_pem_server	boolean	false	none	Determines whether to execute the provided script on the PEM host or the host where the agent is monitoring the object.
» script_code	string	false	none	The shell or batch script to be executed. Please read the documentation for detailed information.
» submit_to_nagios	boolean	false	none	Determines whether to send notification to the nagios when the state of the alert changes.
» override_default_config	boolean	false	none	Determines whether to override the default webhook configuration.
» send_notification	boolean	false	none	Determines whether to enable the webhook send notification.
» low_webhook_ids	[integer]	false	none	Webhook IDs for which low alerts are configured
» med_webhook_ids	[integer]	false	none	Webhook IDs for which medium alerts are configured
» high_webhook_ids	[integer]	false	none	Webhook IDs for which high alerts are configured
» cleared_webhook_ids	[integer]	false	none	Webhook IDs for which cleared alerts are configured

Enumerated Values

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%)

Note

This operation does not require authentication

post_*alert_config_server*{server_id}

Code samples

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v16/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/v16/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,
```

(continues on next page)

(continued from previous page)

```
  "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/v16/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}`

Creates `Alert` for server level.

Body parameter

```
{
  "alert_name": "string",
  "alert_template": 0,
  "description": "string",
  "enabled": true,
  "history_retention": 0,
```

(continues on next page)

(continued from previous page)


```
  "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": []
}
```


Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
X-Auth-Token	header	string	true	Token of authorization
body	body	*AlertPayload*	true	none

Example responses

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Status 200	None

Response Schema

> **Note**
>
> This operation does not require authentication

get_*alert_config_server*{server_id}_{alert_id}

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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/v16/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/v16/alert/config/server/{server_id}/{alert_id},
{
  method: GET,

  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

```
GET /alert/config/server/{server_id}/{alert_id}
```

Gets `Alerts` for server level by server_id and alert_id.

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
alert_id	path	integer	true	Alert ID
X-Auth-Token	header	string	true	Token of authorization

Example responses


```
{
  "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,
```


(continues on next page)

(continued from previous page)

```
  "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
  ]
}
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	*About alert threads*

Note

This operation does not require authentication

put_*alert_config_server*{server_id}_{alert_id}

Code samples

```
#  You can also use wget

curl -X PUT https://PEM-SERVER-IP/pem/api/v16/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/v16/alert/config/server/{server_id}/
↪{alert_id}, headers = headers)
```

(continues on next page)

(continued from previous page)

```
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,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v16/alert/config/server/{server_id}/{alert_id},
{
  method: PUT,
  body: inputBody,
  headers: headers
})
```

(continues on next page)

(continued from previous page)

```
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

`PUT /alert/config/server/{server_id}/{alert_id}`

Updates `Alert` for server level.

Body parameter

```
{
  "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": []
}
```

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
alert_id	path	integer	true	Alert ID
X-Auth-Token	header	string	true	Token of authorization
body	body	*AlertPayload*	true	none

Example responses

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Status 200	None

Response Schema

Note

This operation does not require authentication

delete_*alert_config_server*{server_id}_{alert_id}

Code samples

```
#  You can also use wget

curl -X DELETE https://PEM-SERVER-IP/pem/api/v16/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/v16/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/v16/alert/config/server/{server_id}/{alert_id},
```

(continues on next page)

(continued from previous page)

```
{
  method: DELETE,

  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

```
DELETE /alert/config/server/{server_id}/{alert_id}
```

Deletes the `Alert` object.

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
alert_id	path	integer	true	Alert ID
X-Auth-Token	header	string	true	Token of authorization

Example responses

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful deletion	None

Response Schema

Note

This operation does not require authentication

get_*alert_config_server*{server_id}*database*{database_name}

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/alert/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/v16/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/v16/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}`

Gets `Alerts` for database level by server_id and database_name.

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
database_name	path	string	true	Database Name
X-Auth-Token	header	string	true	Token of authorization

Example responses

```
[
  {
    "id": 0,
    "alert_name": "string",
    "alert_template": 0,
    "description": "string",
    "enabled": true,
    "history_retention": 0,
    "frequency_min": 0,
```

(continues on next page)

(continued from previous page)


```
    "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
    ]
  }
]
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	Inline

Response Schema

Status Code **200**

Name	Type	Required	Restrictions	Description
anonymous	*Query information*]	false	none	none
» id	integer	false	none	Alert ID. NOTE: This property is not applicable for a POST request.
» alert_name	string	false	none	Name of the alert. NOTE: This property is a mandatory parameter for a POST request.
» alert_template	integer	false	none	ID of the alert template. This property must be a positive integer value. NOTE: It is a mandatory parameter for a POST request.
» description	string	false	none	none
» enabled	boolean	false	none	Determines whether this alert is enabled/disabled. NOTE: This property is a mandatory parameter for a POST request.
» history_retention	integer	false	none	Specifies how many days to keep the history of this alert. Possible values are from 1 to 99999. NOTE: This property is a mandatory parameter for a POST request.
» frequency_min	integer	false	none	Defines in minutes how frequently the alert must be checked on a PEM database server. Possible values are from 1 to 65534. NOTE: This property is a mandatory parameter for a POST request.
» auto_created	boolean	false	none	Specifies whether the alert is auto-created.

continues on next page

Table 185 – continued from previous page

Name	Type	Required	Restrictions	Description
» template_id	string	false	none	Defines the alert template.
» params	*Query information*]	false	none	List of parameter/value pairs to be used by the alert template for checking alerts. NOTE: The alert template requires the all parameter to be defined.
»» paramname	string	false	none	Name of the parameter. NOTE: This property must present on the alert template parameter list.Example:
»» paramvalue	string	false	none	Value for the given parameter
» param_names	[string]	false	none	Defines the name of parameters.
» param_types	string	false	none	Defines the type of parameters. Example: INTEGER/STRING..
» params_units	[string]	false	none	Defines the unit used for parameters.Example: 'DAYS','True or False','Hours','Minutes' etc
» threshold_unit	string	false	none	Units used to calculate the threshold, such as MB/kB/seconds/days/... This value is only used for the GUI.
» operator	string	false	none	Defines the operator for checking threshold values. Valid strings are ">" and "<". NOTE: This property is a mandatory parameter for a POST request.
» low_threshold_value	number	false	none	Threshold value for triggering the low alert. This property can be any valid integer/float value. NOTE: It is a mandatory parameter for the POST request.
» medium_threshold_value	number	false	none	Threshold value for triggering the medium alert. This property can be any valid integer/float value. NOTE: It is a mandatory parameter for the POST request.

continues on next page

Table 185 – continued from previous page

Name	Type	Required	Restrictions	Description
» high_threshold_value	number	false	none	Threshold value for triggering the high alert. It can be any valid integer/float value. NOTE: It is a mandatory parameter for the POST request.
» send_email	boolean	false	none	Determines whether to send an email when the state of the alert changes.
» all_alert_enable	boolean	false	none	Determines whether to send an email notification to the group specified by the parameter 'email_group_id' when the state of the alert changes.
» email_group_id	integer	false	none	Defines the email group to which email notification are sent when the state of the alert changes.
» low_alert_enable	boolean	false	none	Determines whether to send an email notification to the group specified by the parameter 'low_email_group_id' when the state of the alert is set to LOW.
» low_email_group_id	integer	false	none	Defines the email group to which email notification are sent when the state of the alert changes to LOW.
» med_alert_enable	boolean	false	none	Determines whether to send an email notification to the group specified by the parameter 'medium_email_group_id' when the state of the alert is set to MEDIUM.
» med_email_group_id	integer	false	none	Defines the email group to which email notification are sent when state of the alert changes to MEDIUM.
» high_alert_enable	boolean	false	none	Determines whether to send an email notification to the group specified by the parameter 'high_email_group_id' when the state of the alert is set to HIGH.

continues on next page

Table 185 – continued from previous page

Name	Type	Required	Restrictions	Description
» high_email_group_id	integer	false	none	Defines the email group to which email notification are sent when the state of the alert changes to HIGH.
» send_trap	boolean	false	none	Determines whether to send SNMP trap notifications when the state of the alert changes.
» snmp_trap_version	integer	false	none	Defines the SNMP trap version. Supported SNMP trap versions are 1, 2 or 3
» low_send_trap	boolean	false	none	Determines whether to send SNMP trap notifications when the state of the alert changes to LOW.
» med_send_trap	boolean	false	none	Determines whether to send SNMP trap notification when the state of the alert changes to MEDIUM.
» high_send_trap	boolean	false	none	Determines whether to send SNMP trap notification when the state of the alert changes to HIGH.
» cleared_alert_enable	boolean	false	none	Determines whether to mute the notification for 'clear alert'.
» execute_script	boolean	false	none	Determines whether to execute the provided script if the state of the alert changes to LOW/MEDIUM/HIGH.
» execute_script_on_clear	boolean	false	none	Determines whether to execute the provided script if the state of the alert is cleared.
» execute_script_on_pem_server	boolean	false	none	Determines whether to execute the provided script on the PEM host or the host where the agent is monitoring the object.
» script_code	string	false	none	The shell or batch script to be executed. Please read the documentation for detailed information.
» submit_to_nagios	boolean	false	none	Determines whether to send notification to the nagios when the state of the alert changes.

continues on next page

Table 185 – continued from previous page

Name	Type	Required	Restrictions	Description
» override_default_config	boolean	false	none	Determines whether to override the default webhook configuration.
» send_notification	boolean	false	none	Determines whether to enable the webhook send notification.
» low_webhook_ids	[integer]	false	none	Webhook IDs for which low alerts are configured
» med_webhook_ids	[integer]	false	none	Webhook IDs for which medium alerts are configured
» high_webhook_ids	[integer]	false	none	Webhook IDs for which high alerts are configured
» cleared_webhook_ids	[integer]	false	none	Webhook IDs for which cleared alerts are configured

Enumerated Values

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%)

Note

This operation does not require authentication

post_*alert_config_server*{server_id}*database*{database_name}

Code samples

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v16/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
```

(continues on next page)

(continued from previous page)

```
}

r = requests.post(https://PEM-SERVER-IP/pem/api/v16/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,
  "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/v16/alert/config/server/{server_id}/database/
↪{database_name},
```

(continues on next page)

(continued from previous page)

```
{
  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}`

Creates `Alert` for database level.

Body parameter

```
{
  "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": [],
```

(continues on next page)

(continued from previous page)

```
  "cleared_webhook_ids": []
}
```

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
database_name	path	string	true	Database Name
X-Auth-Token	header	string	true	Token of authorization
body	body	*AlertPayload*	true	none

Example responses

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Status 200	None

Response Schema

> **Note**
>
> This operation does not require authentication

get_*alert_config_server*{server_id}*database*{database_name}_{alert_id}

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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/v16/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/v16/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}`

Gets `Alerts` for database level by server_id, database_name, and alert_id.

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
database_name	path	string	true	Database name
alert_id	path	integer	true	Alert ID
X-Auth-Token	header	string	true	Token of authorization

Example responses

```
{
  "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",
```

(continues on next page)

(continued from previous page)


```
  "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
  ]
}
```


Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	*About alert threads*

Note

This operation does not require authentication

put_*alert_config_server*{server_id}*database*{database_name}_{alert_id}

Code samples

```
#  You can also use wget

curl -X PUT https://PEM-SERVER-IP/pem/api/v16/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/v16/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,
  "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,
```

(continues on next page)

(continued from previous page)

```
  "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/v16/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}`

Updates `Alert` for database level.

Body parameter

```
{
  "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,
```

(continues on next page)

(continued from previous page)


```
  "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": []
}
```


Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
database_name	path	string	true	Database name
alert_id	path	integer	true	Alert ID
X-Auth-Token	header	string	true	Token of authorization
body	body	*AlertPayload*	true	none

Example responses

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Status 200	None

Response Schema

Note

This operation does not require authentication

delete_*alert_config_server*{server_id}*database*{database_name}_{alert_id}

Code samples

```
#  You can also use wget

curl -X DELETE https://PEM-SERVER-IP/pem/api/v16/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/v16/alert/config/server/{server_id}/
↪database/{database_name}/{alert_id}, headers = headers)

print(r.json())
```

```
const headers = {
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v16/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}`

Deletes the `Alert` object.

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
database_name	path	string	true	Database name
alert_id	path	integer	true	Alert ID
X-Auth-Token	header	string	true	Token of authorization

Example responses

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful deletion	None

Response Schema

Note

This operation does not require authentication

get_*alert_config_server*{server_id}*database*{database_name}*schema*{schema_name}

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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/v16/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/v16/alert/config/server/{server_id}/database/
↪{database_name}/schema/{schema_name},
{
  method: GET,

  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
```

(continues on next page)

(continued from previous page)

```
    console.log(body);
});
```

`GET /alert/config/server/{server_id}/database/{database_name}/schema/{schema_name}`

Gets `Alerts` for schema level by server_id, database_name, and schema_name.

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
database_name	path	string	true	Database Name
schema_name	path	string	true	Schema Name
X-Auth-Token	header	string	true	Token of authorization

Example responses


```
[
  {
    "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,
```


(continues on next page)

(continued from previous page)


```
    "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
    ]
  }
]
```


Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	Inline

Response Schema

Status Code **200**

Name	Type	Required	Restrictions	Description
anonymous	*Query information*]	false	none	none
» id	integer	false	none	Alert ID. NOTE: This property is not applicable for a POST request.
» alert_name	string	false	none	Name of the alert. NOTE: This property is a mandatory parameter for a POST request.

continues on next page

Table 197 – continued from previous page

Name	Type	Required	Restrictions	Description
» alert_template	integer	false	none	ID of the alert template. This property must be a positive integer value. NOTE: It is a mandatory parameter for a POST request.
» description	string	false	none	none
» enabled	boolean	false	none	Determines whether this alert is enabled/disabled. NOTE: This property is a mandatory parameter for a POST request.
» history_retention	integer	false	none	Specifies how many days to keep the history of this alert. Possible values are from 1 to 99999. NOTE: This property is a mandatory parameter for a POST request.
» frequency_min	integer	false	none	Defines in minutes how frequently the alert must be checked on a PEM database server. Possible values are from 1 to 65534. NOTE: This property is a mandatory parameter for a POST request.
» auto_created	boolean	false	none	Specifies whether the alert is auto-created.
» template_id	string	false	none	Defines the alert template.
» params	*Query information*]	false	none	List of parameter/value pairs to be used by the alert template for checking alerts. NOTE: The alert template requires the all parameter to be defined.
»» paramname	string	false	none	Name of the parameter. NOTE: This property must present on the alert template parameter list.Example:
»» paramvalue	string	false	none	Value for the given parameter
» param_names	[string]	false	none	Defines the name of parameters.
» param_types	string	false	none	Defines the type of parameters. Example: INTEGER/STRING..

continues on next page

Table 197 – continued from previous page

Name	Type	Required	Restrictions	Description
» params_units	[string]	false	none	Defines the unit used for parameters.Example: 'DAYS','True or False','Hours','Minutes' etc
» threshold_unit	string	false	none	Units used to calculate the threshold, such as MB/kB/seconds/days/... This value is only used for the GUI.
» operator	string	false	none	Defines the operator for checking threshold values. Valid strings are ">" and "<". NOTE: This property is a mandatory parameter for a POST request.
» low_threshold_value	number	false	none	Threshold value for triggering the low alert. This property can be any valid integer/float value. NOTE: It is a mandatory parameter for the POST request.
» medium_threshold_value	number	false	none	Threshold value for triggering the medium alert. This property can be any valid integer/float value. NOTE: It is a mandatory parameter for the POST request.
» high_threshold_value	number	false	none	Threshold value for triggering the high alert. It can be any valid integer/float value. NOTE: It is a mandatory parameter for the POST request.
» send_email	boolean	false	none	Determines whether to send an email when the state of the alert changes.
» all_alert_enable	boolean	false	none	Determines whether to send an email notification to the group specified by the parameter 'email_group_id' when the state of the alert changes.
» email_group_id	integer	false	none	Defines the email group to which email notification are sent when the state of the alert changes.

continues on next page

Table 197 – continued from previous page

Name	Type	Required	Restrictions	Description
» low_alert_enable	boolean	false	none	Determines whether to send an email notification to the group specified by the parameter 'low_email_group_id' when the state of the alert is set to LOW.
» low_email_group_id	integer	false	none	Defines the email group to which email notification are sent when the state of the alert changes to LOW.
» med_alert_enable	boolean	false	none	Determines whether to send an email notification to the group specified by the parameter 'medium_email_group_id' when the state of the alert is set to MEDIUM.
» med_email_group_id	integer	false	none	Defines the email group to which email notification are sent when state of the alert changes to MEDIUM.
» high_alert_enable	boolean	false	none	Determines whether to send an email notification to the group specified by the parameter 'high_email_group_id' when the state of the alert is set to HIGH.
» high_email_group_id	integer	false	none	Defines the email group to which email notification are sent when the state of the alert changes to HIGH.
» send_trap	boolean	false	none	Determines whether to send SNMP trap notifications when the state of the alert changes.
» snmp_trap_version	integer	false	none	Defines the SNMP trap version. Supported SNMP trap versions are 1, 2 or 3
» low_send_trap	boolean	false	none	Determines whether to send SNMP trap notifications when the state of the alert changes to LOW.
» med_send_trap	boolean	false	none	Determines whether to send SNMP trap notification when the state of the alert changes to MEDIUM.

continues on next page

Table 197 – continued from previous page

Name	Type	Required	Restrictions	Description
» high_send_trap	boolean	false	none	Determines whether to send SNMP trap notification when the state of the alert changes to HIGH.
» cleared_alert_enable	boolean	false	none	Determines whether to mute the notification for 'clear alert'.
» execute_script	boolean	false	none	Determines whether to execute the provided script if the state of the alert changes to LOW/MEDIUM/HIGH.
» execute_script_on_clear	boolean	false	none	Determines whether to execute the provided script if the state of the alert is cleared.
» execute_script_on_pem_server	boolean	false	none	Determines whether to execute the provided script on the PEM host or the host where the agent is monitoring the object.
» script_code	string	false	none	The shell or batch script to be executed. Please read the documentation for detailed information.
» submit_to_nagios	boolean	false	none	Determines whether to send notification to the nagios when the state of the alert changes.
» override_default_config	boolean	false	none	Determines whether to override the default webhook configuration.
» send_notification	boolean	false	none	Determines whether to enable the webhook send notification.
» low_webhook_ids	[integer]	false	none	Webhook IDs for which low alerts are configured
» med_webhook_ids	[integer]	false	none	Webhook IDs for which medium alerts are configured
» high_webhook_ids	[integer]	false	none	Webhook IDs for which high alerts are configured
» cleared_webhook_ids	[integer]	false	none	Webhook IDs for which cleared alerts are configured

Enumerated Values

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%)

Note

This operation does not require authentication

post_*alert_config_server*{server_id}*database*{database_name}*schema*{schema_name}

Code samples

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v16/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,
  X-Auth-Token: string
}

r = requests.post(https://PEM-SERVER-IP/pem/api/v16/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,
```

(continues on next page)

(continued from previous page)

```
  "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/v16/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}`

Creates `Alert` for schema level.

Body parameter

```
{
  "alert_name": "string",
  "alert_template": 0,
  "description": "string",
```

(continues on next page)

(continued from previous page)


```
  "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": []
}
```


Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
database_name	path	string	true	Database Name
schema_name	path	string	true	Schema Name
X-Auth-Token	header	string	true	Token of authorization
body	body	*AlertPayload*	true	none

Example responses

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Status 200	None

Response Schema

Note

This operation does not require authentication

get_*alert_config_server*{server_id}*database*{database_name}*schema*{schema_name}_{alert_id}

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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/v16/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/v16/alert/config/server/{server_id}/database/
↪{database_name}/schema/{schema_name}/{alert_id},
{
  method: GET,

  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
```

(continues on next page)

(continued from previous page)

```
    console.log(body);
});
```

`GET /alert/config/server/{server_id}/database/{database_name}/schema/{schema_name}/{alert_id}`

Gets `Alerts` for schema level by server_id, database_name, schema_name, and alert_id.

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
database_name	path	string	true	Database name
schema_name	path	string	true	Schema name
alert_id	path	integer	true	Alert ID
X-Auth-Token	header	string	true	Token of authorization

Example responses


```
{
  "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,
```


(continues on next page)

(continued from previous page)


```
  "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
  ]
}
```


Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	*About alert threads*

Note

This operation does not require authentication

put_*alert_config_server*{server_id}*database*{database_name}*schema*{schema_name}_{alert_id}

Code samples

```
#  You can also use wget

curl -X PUT https://PEM-SERVER-IP/pem/api/v16/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/v16/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,
  "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
```

(continues on next page)

(continued from previous page)

```
};

fetch(https://PEM-SERVER-IP/pem/api/v16/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}`

Updates `Alert` for schema level.

Body parameter

```
{
  "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,
```

(continues on next page)

(continued from previous page)

```
  "override_default_config": true,
  "send_notification": false,
  "low_webhook_ids": [],
  "med_webhook_ids": [],
  "high_webhook_ids": [],
  "cleared_webhook_ids": []
}
```

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
database_name	path	string	true	Database name
schema_name	path	string	true	Schema name
alert_id	path	integer	true	Alert ID
X-Auth-Token	header	string	true	Token of authorization
body	body	*AlertPayload*	true	none

Example responses

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Status 200	None

Response Schema

Note

This operation does not require authentication

delete_*alert_config_server*{server_id}*database*{database_name}*schema*{schema_name}_{alert_id}

Code samples

```
#  You can also use wget

curl -X DELETE https://PEM-SERVER-IP/pem/api/v16/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
```

(continues on next page)

(continued from previous page)

```
}

r = requests.delete(https://PEM-SERVER-IP/pem/api/v16/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/v16/alert/config/server/{server_id}/database/
↪{database_name}/schema/{schema_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}/{alert_id}`

Deletes the `Alert` object.

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
database_name	path	string	true	Database name
schema_name	path	string	true	Schema name
alert_id	path	integer	true	Alert ID
X-Auth-Token	header	string	true	Token of authorization

Example responses

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful deletion	None

Response Schema

> **Note**
>
> This operation does not require authentication

get_*alert_config_server*{server_id}*database*{database_name}*schema*{schema_name}*table*{table_name}

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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/v16/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/v16/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}
```

Gets `Alerts` for table level by server_id, database_name, schema_name, and table_name.

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
database_name	path	string	true	Database name
schema_name	path	string	true	Schema name
table_name	path	string	true	Table name
X-Auth-Token	header	string	true	Token of authorization

Example responses


```
[
  {
    "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,
```


(continues on next page)

(continued from previous page)

```
    "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
    ]
  }
]
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	Inline

Response Schema

Status Code **200**

Name	Type	Required	Restrictions	Description
anonymous	*Query information*]	false	none	none
» id	integer	false	none	Alert ID. NOTE: This property is not applicable for a POST request.
» alert_name	string	false	none	Name of the alert. NOTE: This property is a mandatory parameter for a POST request.
» alert_template	integer	false	none	ID of the alert template. This property must be a positive integer value. NOTE: It is a mandatory parameter for a POST request.
» description	string	false	none	none

continues on next page

Table 209 – continued from previous page

Name	Type	Required	Restrictions	Description
» enabled	boolean	false	none	Determines whether this alert is enabled/disabled. NOTE: This property is a mandatory parameter for a POST request.
» history_retention	integer	false	none	Specifies how many days to keep the history of this alert. Possible values are from 1 to 99999. NOTE: This property is a mandatory parameter for a POST request.
» frequency_min	integer	false	none	Defines in minutes how frequently the alert must be checked on a PEM database server. Possible values are from 1 to 65534. NOTE: This property is a mandatory parameter for a POST request.
» auto_created	boolean	false	none	Specifies whether the alert is auto-created.
» template_id	string	false	none	Defines the alert template.
» params	*Query information*]	false	none	List of parameter/value pairs to be used by the alert template for checking alerts. NOTE: The alert template requires the all parameter to be defined.
»» paramname	string	false	none	Name of the parameter. NOTE: This property must present on the alert template parameter list.Example:
»» paramvalue	string	false	none	Value for the given parameter
» param_names	[string]	false	none	Defines the name of parameters.
» param_types	string	false	none	Defines the type of parameters. Example: INTEGER/STRING..
» params_units	[string]	false	none	Defines the unit used for parameters.Example: 'DAYS','True or False','Hours','Minutes' etc

continues on next page

Table 209 – continued from previous page

Name	Type	Required	Restrictions	Description
» threshold_unit	string	false	none	Units used to calculate the threshold, such as MB/kB/seconds/days/... This value is only used for the GUI.
» operator	string	false	none	Defines the operator for checking threshold values. Valid strings are ">" and "<". NOTE: This property is a mandatory parameter for a POST request.
» low_threshold_value	number	false	none	Threshold value for triggering the low alert. This property can be any valid integer/float value. NOTE: It is a mandatory parameter for the POST request.
» medium_threshold_value	number	false	none	Threshold value for triggering the medium alert. This property can be any valid integer/float value. NOTE: It is a mandatory parameter for the POST request.
» high_threshold_value	number	false	none	Threshold value for triggering the high alert. It can be any valid integer/float value. NOTE: It is a mandatory parameter for the POST request.
» send_email	boolean	false	none	Determines whether to send an email when the state of the alert changes.
» all_alert_enable	boolean	false	none	Determines whether to send an email notification to the group specified by the parameter 'email_group_id' when the state of the alert changes.
» email_group_id	integer	false	none	Defines the email group to which email notification are sent when the state of the alert changes.

continues on next page

Table 209 – continued from previous page

Name	Type	Required	Restrictions	Description
» low_alert_enable	boolean	false	none	Determines whether to send an email notification to the group specified by the parameter 'low_email_group_id' when the state of the alert is set to LOW.
» low_email_group_id	integer	false	none	Defines the email group to which email notification are sent when the state of the alert changes to LOW.
» med_alert_enable	boolean	false	none	Determines whether to send an email notification to the group specified by the parameter 'medium_email_group_id' when the state of the alert is set to MEDIUM.
» med_email_group_id	integer	false	none	Defines the email group to which email notification are sent when state of the alert changes to MEDIUM.
» high_alert_enable	boolean	false	none	Determines whether to send an email notification to the group specified by the parameter 'high_email_group_id' when the state of the alert is set to HIGH.
» high_email_group_id	integer	false	none	Defines the email group to which email notification are sent when the state of the alert changes to HIGH.
» send_trap	boolean	false	none	Determines whether to send SNMP trap notifications when the state of the alert changes.
» snmp_trap_version	integer	false	none	Defines the SNMP trap version. Supported SNMP trap versions are 1, 2 or 3
» low_send_trap	boolean	false	none	Determines whether to send SNMP trap notifications when the state of the alert changes to LOW.
» med_send_trap	boolean	false	none	Determines whether to send SNMP trap notification when the state of the alert changes to MEDIUM.

continues on next page

Table 209 – continued from previous page

Name	Type	Required	Restrictions	Description
» high_send_trap	boolean	false	none	Determines whether to send SNMP trap notification when the state of the alert changes to HIGH.
» cleared_alert_enable	boolean	false	none	Determines whether to mute the notification for 'clear alert'.
» execute_script	boolean	false	none	Determines whether to execute the provided script if the state of the alert changes to LOW/MEDIUM/HIGH.
» execute_script_on_clear	boolean	false	none	Determines whether to execute the provided script if the state of the alert is cleared.
» execute_script_on_pem_server	boolean	false	none	Determines whether to execute the provided script on the PEM host or the host where the agent is monitoring the object.
» script_code	string	false	none	The shell or batch script to be executed. Please read the documentation for detailed information.
» submit_to_nagios	boolean	false	none	Determines whether to send notification to the nagios when the state of the alert changes.
» override_default_config	boolean	false	none	Determines whether to override the default webhook configuration.
» send_notification	boolean	false	none	Determines whether to enable the webhook send notification.
» low_webhook_ids	[integer]	false	none	Webhook IDs for which low alerts are configured
» med_webhook_ids	[integer]	false	none	Webhook IDs for which medium alerts are configured
» high_webhook_ids	[integer]	false	none	Webhook IDs for which high alerts are configured
» cleared_webhook_ids	[integer]	false	none	Webhook IDs for which cleared alerts are configured

Enumerated Values

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%)

Note

This operation does not require authentication

post_*alert_config_server*{server_id}*database*{database_name}*schema*{schema_name}*table*{table_name}

Code samples

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v16/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/v16/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",
  "low_threshold_value": 0,
  "medium_threshold_value": 0,
  "high_threshold_value": 0,
  "all_alert_enable": true,
  "email_group_id": 1,
```

(continues on next page)

(continued from previous page)

```
  "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/v16/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) {
    console.log(body);
});
```

`POST /alert/config/server/{server_id}/database/{database_name}/schema/{schema_name}/table/{table_name}`

Creates `Alert` for table level.

Body parameter

```
{
  "alert_name": "string",
  "alert_template": 0,
```

(continues on next page)

(continued from previous page)


```
  "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": []
}
```


Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
database_name	path	string	true	Database name
schema_name	path	string	true	Schema name
table_name	path	string	true	Table name
X-Auth-Token	header	string	true	Token of authorization
body	body	*AlertPayload*	true	none

Example responses

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Status 200	None

Response Schema

Note

This operation does not require authentication

get_*alert_config_server*{server_id}*database*{database_name}*schema*{schema_name}*table*{table_name}_{alert_id

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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/v16/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/v16/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();
```

(continues on next page)

(continued from previous page)

```
}).then(function(body) {
    console.log(body);
});
```

`GET /alert/config/server/{server_id}/database/{database_name}/schema/{schema_name}/table/{table_name}/{alert_id}`

Gets `Alerts` for table level by server_id, database_name, schema_name, table_name, and alert_id.

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
database_name	path	string	true	Database Name
schema_name	path	string	true	Schema name
table_name	path	string	true	Table name
alert_id	path	integer	true	Alert ID
X-Auth-Token	header	string	true	Token of authorization

Example responses


```
{
  "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,
```


(continues on next page)

(continued from previous page)

```
  "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
  ]
}
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	*About alert threads*

Note

This operation does not require authentication

put_*alert_config_server*{server_id}*database*{database_name}*schema*{schema_name}*table*{table_name}_{alert_id

Code samples

```
#  You can also use wget

curl -X PUT https://PEM-SERVER-IP/pem/api/v16/alert/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/table/{table_name}/{alert_id} \
```

(continues on next page)

(continued from previous page)

```
  -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/v16/alert/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/table/{table_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,
  "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": [],
```

(continues on next page)

(continued from previous page)

```
  "cleared_webhook_ids": []
};
const headers = {
  Content-Type:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v16/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}`

Updates `Alert` for table level.

Body parameter

```
{
  "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": [],
```

(continues on next page)

(continued from previous page)

```
  "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": []
}
```

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
database_name	path	string	true	Database name
schema_name	path	string	true	Schema name
table_name	path	string	true	Table name
alert_id	path	integer	true	Alert ID
X-Auth-Token	header	string	true	Token of authorization
body	body	*AlertPayload*	true	none

Example responses

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Status 200	None

Response Schema

> **Note**
>
> This operation does not require authentication

delete_*alert_config_server*{server_id}*database*{database_name}*schema*{schema_name}*table*{table_name}_{alert

Code samples

```
#  You can also use wget

curl -X DELETE https://PEM-SERVER-IP/pem/api/v16/alert/config/server/{server_id}/
```

(continues on next page)

(continued from previous page)

```
↪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/v16/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/v16/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}`

Deletes the `Alert` object.

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
database_name	path	string	true	Database name
schema_name	path	string	true	Schema name
table_name	path	string	true	Table name
alert_id	path	integer	true	Alert ID
X-Auth-Token	header	string	true	Token of authorization

Example responses

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful deletion	None

Response Schema

Note

This operation does not require authentication

get_*alert_config_server*{server_id}*database*{database_name}*schema*{schema_name}*index*{index_name}

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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 = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v16/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/v16/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) {
```

(continues on next page)

(continued from previous page)

```
    console.log(body);
});
```

`GET /alert/config/server/{server_id}/database/{database_name}/schema/{schema_name}/index/{index_name}`

Gets `Alerts` for index level by server_id, database_name, schema_name, and index_name.

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
database_name	path	string	true	Database name
schema_name	path	string	true	Schema name
index_name	path	string	true	Index name
X-Auth-Token	header	string	true	Token of authorization

Example responses


```
[
  {
    "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,
```


(continues on next page)

(continued from previous page)


```
    "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
    ]
  }
]
```


Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	Inline

Response Schema

Status Code **200**

Name	Type	Required	Restrictions	Description
anonymous	*Query information*]	false	none	none
» id	integer	false	none	Alert ID. NOTE: This property is not applicable for a POST request.

continues on next page

Table 221 – continued from previous page

Name	Type	Required	Restrictions	Description
» alert_name	string	false	none	Name of the alert. NOTE: This property is a mandatory parameter for a POST request.
» alert_template	integer	false	none	ID of the alert template. This property must be a positive integer value. NOTE: It is a mandatory parameter for a POST request.
» description	string	false	none	none
» enabled	boolean	false	none	Determines whether this alert is enabled/disabled. NOTE: This property is a mandatory parameter for a POST request.
» history_retention	integer	false	none	Specifies how many days to keep the history of this alert. Possible values are from 1 to 99999. NOTE: This property is a mandatory parameter for a POST request.
» frequency_min	integer	false	none	Defines in minutes how frequently the alert must be checked on a PEM database server. Possible values are from 1 to 65534. NOTE: This property is a mandatory parameter for a POST request.
» auto_created	boolean	false	none	Specifies whether the alert is auto-created.
» template_id	string	false	none	Defines the alert template.
» params	*Query information*]	false	none	List of parameter/value pairs to be used by the alert template for checking alerts. NOTE: The alert template requires the all parameter to be defined.
»» paramname	string	false	none	Name of the parameter. NOTE: This property must present on the alert template parameter list.Example:
»» paramvalue	string	false	none	Value for the given parameter
» param_names	[string]	false	none	Defines the name of parameters.

continues on next page

Table 221 – continued from previous page

Name	Type	Required	Restrictions	Description
» param_types	string	false	none	Defines the type of parameters. Example: INTEGER/STRING..
» params_units	[string]	false	none	Defines the unit used for parameters.Example: 'DAYS','True or False','Hours','Minutes' etc
» threshold_unit	string	false	none	Units used to calculate the threshold, such as MB/kB/seconds/days/... This value is only used for the GUI.
» operator	string	false	none	Defines the operator for checking threshold values. Valid strings are ">" and "<". NOTE: This property is a mandatory parameter for a POST request.
» low_threshold_value	number	false	none	Threshold value for triggering the low alert. This property can be any valid integer/float value. NOTE: It is a mandatory parameter for the POST request.
» medium_threshold_value	number	false	none	Threshold value for triggering the medium alert. This property can be any valid integer/float value. NOTE: It is a mandatory parameter for the POST request.
» high_threshold_value	number	false	none	Threshold value for triggering the high alert. It can be any valid integer/float value. NOTE: It is a mandatory parameter for the POST request.
» send_email	boolean	false	none	Determines whether to send an email when the state of the alert changes.
» all_alert_enable	boolean	false	none	Determines whether to send an email notification to the group specified by the parameter 'email_group_id' when the state of the alert changes.

continues on next page

Table 221 – continued from previous page

Name	Type	Required	Restrictions	Description
» email_group_id	integer	false	none	Defines the email group to which email notification are sent when the state of the alert changes.
» low_alert_enable	boolean	false	none	Determines whether to send an email notification to the group specified by the parameter 'low_email_group_id' when the state of the alert is set to LOW.
» low_email_group_id	integer	false	none	Defines the email group to which email notification are sent when the state of the alert changes to LOW.
» med_alert_enable	boolean	false	none	Determines whether to send an email notification to the group specified by the parameter 'medium_email_group_id' when the state of the alert is set to MEDIUM.
» med_email_group_id	integer	false	none	Defines the email group to which email notification are sent when state of the alert changes to MEDIUM.
» high_alert_enable	boolean	false	none	Determines whether to send an email notification to the group specified by the parameter 'high_email_group_id' when the state of the alert is set to HIGH.
» high_email_group_id	integer	false	none	Defines the email group to which email notification are sent when the state of the alert changes to HIGH.
» send_trap	boolean	false	none	Determines whether to send SNMP trap notifications when the state of the alert changes.
» snmp_trap_version	integer	false	none	Defines the SNMP trap version. Supported SNMP trap versions are 1, 2 or 3
» low_send_trap	boolean	false	none	Determines whether to send SNMP trap notifications when the state of the alert changes to LOW.

continues on next page

Table 221 – continued from previous page

Name	Type	Required	Restrictions	Description
» med_send_trap	boolean	false	none	Determines whether to send SNMP trap notification when the state of the alert changes to MEDIUM.
» high_send_trap	boolean	false	none	Determines whether to send SNMP trap notification when the state of the alert changes to HIGH.
» cleared_alert_enable	boolean	false	none	Determines whether to mute the notification for 'clear alert'.
» execute_script	boolean	false	none	Determines whether to execute the provided script if the state of the alert changes to LOW/MEDIUM/HIGH.
» execute_script_on_clear	boolean	false	none	Determines whether to execute the provided script if the state of the alert is cleared.
» execute_script_on_pem_server	boolean	false	none	Determines whether to execute the provided script on the PEM host or the host where the agent is monitoring the object.
» script_code	string	false	none	The shell or batch script to be executed. Please read the documentation for detailed information.
» submit_to_nagios	boolean	false	none	Determines whether to send notification to the nagios when the state of the alert changes.
» override_default_config	boolean	false	none	Determines whether to override the default webhook configuration.
» send_notification	boolean	false	none	Determines whether to enable the webhook send notification.
» low_webhook_ids	[integer]	false	none	Webhook IDs for which low alerts are configured
» med_webhook_ids	[integer]	false	none	Webhook IDs for which medium alerts are configured
» high_webhook_ids	[integer]	false	none	Webhook IDs for which high alerts are configured
» cleared_webhook_ids	[integer]	false	none	Webhook IDs for which cleared alerts are configured

Enumerated Values

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%)

Note

This operation does not require authentication

post_*alert_config_server*{server_id}*database*{database_name}*schema*{schema_name}*index*{index_name}

Code samples

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v16/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/v16/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,
```

(continues on next page)

(continued from previous page)

```
  "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/v16/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}`

Create `Alert` for index level.

Body parameter

```
{
  "alert_name": "string",
  "alert_template": 0,
```

(continues on next page)

(continued from previous page)


```
  "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": []
}
```


Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
database_name	path	string	true	Database name
schema_name	path	string	true	Schema name
index_name	path	string	true	Index name
X-Auth-Token	header	string	true	Token of authorization
body	body	*AlertPayload*	true	none

Example responses

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Status 200	None

Response Schema

Note

This operation does not require authentication

get_*alert_config_server*{server_id}*database*{database_name}*schema*{schema_name}*index*{index_name}_{alert_i

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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/v16/alert/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/index/{index_name}/{alert_id},␣
↪headers = headers)

print(r.json())
```

```
const headers = {
  Accept:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v16/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();
```

(continues on next page)

(continued from previous page)

```
}).then(function(body) {
    console.log(body);
});
```

`GET /alert/config/server/{server_id}/database/{database_name}/schema/{schema_name}/index/{index_name}/{alert_id}`

Gets `Alerts` for index level by server_id, database_name, schema_name, index_name, and alert_id.

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
database_name	path	string	true	Database name
schema_name	path	string	true	Schema name
index_name	path	string	true	Index name
alert_id	path	integer	true	Alert ID
X-Auth-Token	header	string	true	Token of authorization

Example responses


```
{
  "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,
```


(continues on next page)

(continued from previous page)


```
  "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
  ]
}
```


Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	*About alert threads*

Note

This operation does not require authentication

put_*alert_config_server*{server_id}*database*{database_name}*schema*{schema_name}*index*{index_name}_{alert_i

Code samples

```
#  You can also use wget

curl -X PUT https://PEM-SERVER-IP/pem/api/v16/alert/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/index/{index_name}/{alert_id} \
```

(continues on next page)

(continued from previous page)

```
  -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/v16/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,
  "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": [],
```

(continues on next page)

(continued from previous page)

```
  "cleared_webhook_ids": []
};
const headers = {
  Content-Type:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v16/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);
});
```

`PUT /alert/config/server/{server_id}/database/{database_name}/schema/{schema_name}/index/{index_name}/{alert_id}`

Updates `Alert` for index level.

Body parameter

```
{
  "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": [],
```

(continues on next page)

(continued from previous page)

```
  "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": []
}
```

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
database_name	path	string	true	Database name
schema_name	path	string	true	Schema name
index_name	path	string	true	Index name
alert_id	path	integer	true	Alert ID
X-Auth-Token	header	string	true	Token of authorization
body	body	*AlertPayload*	true	none

Example responses

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Status 200	None

Response Schema

Note

This operation does not require authentication

delete_*alert_config_server*{server_id}*database*{database_name}*schema*{schema_name}*index*{index_name}_{ale

Code samples

```
#  You can also use wget

curl -X DELETE https://PEM-SERVER-IP/pem/api/v16/alert/config/server/{server_id}/
```

(continues on next page)

(continued from previous page)

```
↪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/v16/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/v16/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}`

Deletes the `Alert` object.

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
database_name	path	string	true	Database name
schema_name	path	string	true	Schema name
index_name	path	string	true	Index name
alert_id	path	integer	true	Alert ID
X-Auth-Token	header	string	true	Token of authorization

Example responses

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful deletion	None

Response Schema

Note

This operation does not require authentication

get_*alert_config_server*{server_id}*database*{database_name}*schema*{schema_name}*sequence*{sequence_name}

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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/v16/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/v16/alert/config/server/{server_id}/database/
↪{database_name}/schema/{schema_name}/sequence/{sequence_name},
{
  method: GET,

  headers: headers
})
.then(function(res) {
    return res.json();
```

(continues on next page)

(continued from previous page)

```
}).then(function(body) {
    console.log(body);
});
```

`GET /alert/config/server/{server_id}/database/{database_name}/schema/{schema_name}/sequence/{sequence_name}`

Gets `Alerts` for sequence level by server_id, database_name, schema_name, and sequence_name.

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
database_name	path	string	true	Database name
schema_name	path	string	true	Schema name
sequence_name	path	string	true	Sequence name
X-Auth-Token	header	string	true	Token of authorization

Example responses


```
[
  {
    "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,
```


(continues on next page)

(continued from previous page)


```
    "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
    ]
  }
]
```


Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	Inline

Response Schema

Status Code **200**

Name	Type	Required	Restrictions	Description
anonymous	*Query information*]	false	none	none
» id	integer	false	none	Alert ID. NOTE: This property is not applicable for a POST request.

continues on next page

Table 233 – continued from previous page

Name	Type	Required	Restrictions	Description
» alert_name	string	false	none	Name of the alert. NOTE: This property is a mandatory parameter for a POST request.
» alert_template	integer	false	none	ID of the alert template. This property must be a positive integer value. NOTE: It is a mandatory parameter for a POST request.
» description	string	false	none	none
» enabled	boolean	false	none	Determines whether this alert is enabled/disabled. NOTE: This property is a mandatory parameter for a POST request.
» history_retention	integer	false	none	Specifies how many days to keep the history of this alert. Possible values are from 1 to 99999. NOTE: This property is a mandatory parameter for a POST request.
» frequency_min	integer	false	none	Defines in minutes how frequently the alert must be checked on a PEM database server. Possible values are from 1 to 65534. NOTE: This property is a mandatory parameter for a POST request.
» auto_created	boolean	false	none	Specifies whether the alert is auto-created.
» template_id	string	false	none	Defines the alert template.
» params	*Query information*]	false	none	List of parameter/value pairs to be used by the alert template for checking alerts. NOTE: The alert template requires the all parameter to be defined.
»» paramname	string	false	none	Name of the parameter. NOTE: This property must present on the alert template parameter list.Example:
»» paramvalue	string	false	none	Value for the given parameter
» param_names	[string]	false	none	Defines the name of parameters.

continues on next page

Table 233 – continued from previous page

Name	Type	Required	Restrictions	Description
» param_types	string	false	none	Defines the type of parameters. Example: INTEGER/STRING..
» params_units	[string]	false	none	Defines the unit used for parameters.Example: 'DAYS','True or False','Hours','Minutes' etc
» threshold_unit	string	false	none	Units used to calculate the threshold, such as MB/kB/seconds/days/... This value is only used for the GUI.
» operator	string	false	none	Defines the operator for checking threshold values. Valid strings are ">" and "<". NOTE: This property is a mandatory parameter for a POST request.
» low_threshold_value	number	false	none	Threshold value for triggering the low alert. This property can be any valid integer/float value. NOTE: It is a mandatory parameter for the POST request.
» medium_threshold_value	number	false	none	Threshold value for triggering the medium alert. This property can be any valid integer/float value. NOTE: It is a mandatory parameter for the POST request.
» high_threshold_value	number	false	none	Threshold value for triggering the high alert. It can be any valid integer/float value. NOTE: It is a mandatory parameter for the POST request.
» send_email	boolean	false	none	Determines whether to send an email when the state of the alert changes.
» all_alert_enable	boolean	false	none	Determines whether to send an email notification to the group specified by the parameter 'email_group_id' when the state of the alert changes.

continues on next page

Table 233 – continued from previous page

Name	Type	Required	Restrictions	Description
» email_group_id	integer	false	none	Defines the email group to which email notification are sent when the state of the alert changes.
» low_alert_enable	boolean	false	none	Determines whether to send an email notification to the group specified by the parameter 'low_email_group_id' when the state of the alert is set to LOW.
» low_email_group_id	integer	false	none	Defines the email group to which email notification are sent when the state of the alert changes to LOW.
» med_alert_enable	boolean	false	none	Determines whether to send an email notification to the group specified by the parameter 'medium_email_group_id' when the state of the alert is set to MEDIUM.
» med_email_group_id	integer	false	none	Defines the email group to which email notification are sent when state of the alert changes to MEDIUM.
» high_alert_enable	boolean	false	none	Determines whether to send an email notification to the group specified by the parameter 'high_email_group_id' when the state of the alert is set to HIGH.
» high_email_group_id	integer	false	none	Defines the email group to which email notification are sent when the state of the alert changes to HIGH.
» send_trap	boolean	false	none	Determines whether to send SNMP trap notifications when the state of the alert changes.
» snmp_trap_version	integer	false	none	Defines the SNMP trap version. Supported SNMP trap versions are 1, 2 or 3
» low_send_trap	boolean	false	none	Determines whether to send SNMP trap notifications when the state of the alert changes to LOW.

continues on next page

Table 233 – continued from previous page

Name	Type	Required	Restrictions	Description
» med_send_trap	boolean	false	none	Determines whether to send SNMP trap notification when the state of the alert changes to MEDIUM.
» high_send_trap	boolean	false	none	Determines whether to send SNMP trap notification when the state of the alert changes to HIGH.
» cleared_alert_enable	boolean	false	none	Determines whether to mute the notification for 'clear alert'.
» execute_script	boolean	false	none	Determines whether to execute the provided script if the state of the alert changes to LOW/MEDIUM/HIGH.
» execute_script_on_clear	boolean	false	none	Determines whether to execute the provided script if the state of the alert is cleared.
» execute_script_on_pem_server	boolean	false	none	Determines whether to execute the provided script on the PEM host or the host where the agent is monitoring the object.
» script_code	string	false	none	The shell or batch script to be executed. Please read the documentation for detailed information.
» submit_to_nagios	boolean	false	none	Determines whether to send notification to the nagios when the state of the alert changes.
» override_default_config	boolean	false	none	Determines whether to override the default webhook configuration.
» send_notification	boolean	false	none	Determines whether to enable the webhook send notification.
» low_webhook_ids	[integer]	false	none	Webhook IDs for which low alerts are configured
» med_webhook_ids	[integer]	false	none	Webhook IDs for which medium alerts are configured
» high_webhook_ids	[integer]	false	none	Webhook IDs for which high alerts are configured
» cleared_webhook_ids	[integer]	false	none	Webhook IDs for which cleared alerts are configured

Enumerated Values

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%)

Note

This operation does not require authentication

post_*alert_config_server*{server_id}*database*{database_name}*schema*{schema_name}*sequence*{sequence_name

Code samples

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v16/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/v16/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,
```

(continues on next page)

(continued from previous page)

```
  "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/v16/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}`

Creates `Alert` for sequence level.

Body parameter

```
{
  "alert_name": "string",
```

(continues on next page)

(continued from previous page)


```
  "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": []
}
```


Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
database_name	path	string	true	Database name
schema_name	path	string	true	Schema name
sequence_name	path	string	true	Sequence Name
X-Auth-Token	header	string	true	Token of authorization
body	body	*AlertPayload*	true	none

Example responses

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Status 200	None

Response Schema

Note

This operation does not require authentication

get_*alert_config_server*{server_id}*database*{database_name}*schema*{schema_name}*sequence*{sequence_name}

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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/v16/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
};

fetch(https://PEM-SERVER-IP/pem/api/v16/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();
```

(continues on next page)

(continued from previous page)

```
}).then(function(body) {
    console.log(body);
});
```

`GET /alert/config/server/{server_id}/database/{database_name}/schema/{schema_name}/sequence/{sequence_name}/{alert_id}`

Gets `Alerts` for sequence level by server_id, database_name, schema_name, sequence_name, and alert_id.

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
database_name	path	string	true	Database name
schema_name	path	string	true	Schema name
sequence_name	path	string	true	Sequence name
alert_id	path	integer	true	Alert ID
X-Auth-Token	header	string	true	Token of authorization

Example responses


```
{
  "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,
```


(continues on next page)

(continued from previous page)

```
  "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
  ]
}
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	*About alert threads*

Note

This operation does not require authentication

put_*alert_config_server*{server_id}*database*{database_name}*schema*{schema_name}*sequence*{sequence_name}

Code samples

```
#  You can also use wget

curl -X PUT https://PEM-SERVER-IP/pem/api/v16/alert/config/server/{server_id}/
↪database/{database_name}/schema/{schema_name}/sequence/{sequence_name}/{alert_id} \
```

(continues on next page)

(continued from previous page)

```
  -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/v16/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,
  "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": [],
```

(continues on next page)

(continued from previous page)

```
  "cleared_webhook_ids": []
};
const headers = {
  Content-Type:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v16/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}`

Updates `Alert` for sequence level.

Body parameter

```
{
  "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": [],
```

(continues on next page)

(continued from previous page)

```
  "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": []
}
```

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
database_name	path	string	true	Database name
schema_name	path	string	true	Schema name
sequence_name	path	string	true	Sequence name
alert_id	path	integer	true	Alert ID
X-Auth-Token	header	string	true	Token of authorization
body	body	*AlertPayload*	true	none

Example responses

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Status 200	None

Response Schema

Note

This operation does not require authentication

delete_*alert_config_server*{server_id}*database*{database_name}*schema*{schema_name}*sequence*{sequence_nar

Code samples

```
#  You can also use wget

curl -X DELETE https://PEM-SERVER-IP/pem/api/v16/alert/config/server/{server_id}/
```

(continues on next page)

(continued from previous page)

```
↪database/{database_name}/schema/{schema_name}/sequence/{sequence_name}/{alert_id} \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  X-Auth-Token: string
}

r = requests.delete(https://PEM-SERVER-IP/pem/api/v16/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/v16/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}`

Delete the `Alert` object.

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
database_name	path	string	true	Database name
schema_name	path	string	true	Schema name
sequence_name	path	string	true	Sequence name
alert_id	path	integer	true	Alert ID
X-Auth-Token	header	string	true	Token of authorization

Example responses

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful deletion	None

Response Schema

Note

This operation does not require authentication

get_*alert_config_server*{server_id}*database*{database_name}*schema*{schema_name}*function*{function_name}*arg*

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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/v16/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(https://PEM-SERVER-IP/pem/api/v16/alert/config/server/{server_id}/database/
↪{database_name}/schema/{schema_name}/function/{function_name}/args/{function_
↪arguments}/,
{
  method: GET,

  headers: headers
})
```

(continues on next page)

(continued from previous page)

```
.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}/
```

Gets `Alerts` for function level by server_id, database_name, schema_name, and function_name.

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
database_name	path	string	true	Database name
schema_name	path	string	true	Schema name
function_name	path	string	true	Function name
function_arguments	path	string	true	Comma-separated list of function arguments.
X-Auth-Token	header	string	true	Token of authorization

Detailed descriptions

function_arguments: Comma-separated list of function arguments.

Note: Provide space if no function argument

Example responses


```
[
  {
    "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,
```


(continues on next page)

(continued from previous page)


```
    "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
    ]
  }
]
```


Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	Inline

Response Schema

Status Code **200**

Name	Type	Required	Restrictions	Description
anonymous	*Query information*]	false	none	none
» id	integer	false	none	Alert ID. NOTE: This property is not applicable for a POST request.
» alert_name	string	false	none	Name of the alert. NOTE: This property is a mandatory parameter for a POST request.
» alert_template	integer	false	none	ID of the alert template. This property must be a positive integer value. NOTE: It is a mandatory parameter for a POST request.
» description	string	false	none	none
» enabled	boolean	false	none	Determines whether this alert is enabled/disabled. NOTE: This property is a mandatory parameter for a POST request.
» history_retention	integer	false	none	Specifies how many days to keep the history of this alert. Possible values are from 1 to 99999. NOTE: This property is a mandatory parameter for a POST request.
» frequency_min	integer	false	none	Defines in minutes how frequently the alert must be checked on a PEM database server. Possible values are from 1 to 65534. NOTE: This property is a mandatory parameter for a POST request.
» auto_created	boolean	false	none	Specifies whether the alert is auto-created.
» template_id	string	false	none	Defines the alert template.
» params	*Query information*]	false	none	List of parameter/value pairs to be used by the alert template for checking alerts. NOTE: The alert template requires the all parameter to be defined.

continues on next page

Table 245 – continued from previous page

Name	Type	Required	Restrictions	Description
»» paramname	string	false	none	Name of the parameter. NOTE: This property must present on the alert template parameter list.Example:
»» paramvalue	string	false	none	Value for the given parameter
» param_names	[string]	false	none	Defines the name of parameters.
» param_types	string	false	none	Defines the type of parameters. Example: INTEGER/STRING..
» params_units	[string]	false	none	Defines the unit used for parameters.Example: 'DAYS','True or False','Hours','Minutes' etc
» threshold_unit	string	false	none	Units used to calculate the threshold, such as MB/kB/seconds/days/... This value is only used for the GUI.
» operator	string	false	none	Defines the operator for checking threshold values. Valid strings are ">" and "<". NOTE: This property is a mandatory parameter for a POST request.
» low_threshold_value	number	false	none	Threshold value for triggering the low alert. This property can be any valid integer/float value. NOTE: It is a mandatory parameter for the POST request.
» medium_threshold_value	number	false	none	Threshold value for triggering the medium alert. This property can be any valid integer/float value. NOTE: It is a mandatory parameter for the POST request.
» high_threshold_value	number	false	none	Threshold value for triggering the high alert. It can be any valid integer/float value. NOTE: It is a mandatory parameter for the POST request.

continues on next page

Table 245 – continued from previous page

Name	Type	Required	Restrictions	Description
» send_email	boolean	false	none	Determines whether to send an email when the state of the alert changes.
» all_alert_enable	boolean	false	none	Determines whether to send an email notification to the group specified by the parameter 'email_group_id' when the state of the alert changes.
» email_group_id	integer	false	none	Defines the email group to which email notification are sent when the state of the alert changes.
» low_alert_enable	boolean	false	none	Determines whether to send an email notification to the group specified by the parameter 'low_email_group_id' when the state of the alert is set to LOW.
» low_email_group_id	integer	false	none	Defines the email group to which email notification are sent when the state of the alert changes to LOW.
» med_alert_enable	boolean	false	none	Determines whether to send an email notification to the group specified by the parameter 'medium_email_group_id' when the state of the alert is set to MEDIUM.
» med_email_group_id	integer	false	none	Defines the email group to which email notification are sent when state of the alert changes to MEDIUM.
» high_alert_enable	boolean	false	none	Determines whether to send an email notification to the group specified by the parameter 'high_email_group_id' when the state of the alert is set to HIGH.
» high_email_group_id	integer	false	none	Defines the email group to which email notification are sent when the state of the alert changes to HIGH.

continues on next page

Table 245 – continued from previous page

Name	Type	Required	Restrictions	Description
» send_trap	boolean	false	none	Determines whether to send SNMP trap notifications when the state of the alert changes.
» snmp_trap_version	integer	false	none	Defines the SNMP trap version. Supported SNMP trap versions are 1, 2 or 3
» low_send_trap	boolean	false	none	Determines whether to send SNMP trap notifications when the state of the alert changes to LOW.
» med_send_trap	boolean	false	none	Determines whether to send SNMP trap notification when the state of the alert changes to MEDIUM.
» high_send_trap	boolean	false	none	Determines whether to send SNMP trap notification when the state of the alert changes to HIGH.
» cleared_alert_enable	boolean	false	none	Determines whether to mute the notification for 'clear alert'.
» execute_script	boolean	false	none	Determines whether to execute the provided script if the state of the alert changes to LOW/MEDIUM/HIGH.
» execute_script_on_clear	boolean	false	none	Determines whether to execute the provided script if the state of the alert is cleared.
» execute_script_on_pem_server	boolean	false	none	Determines whether to execute the provided script on the PEM host or the host where the agent is monitoring the object.
» script_code	string	false	none	The shell or batch script to be executed. Please read the documentation for detailed information.
» submit_to_nagios	boolean	false	none	Determines whether to send notification to the nagios when the state of the alert changes.
» override_default_config	boolean	false	none	Determines whether to override the default webhook configuration.

continues on next page

Table 245 – continued from previous page

Name	Type	Required	Restrictions	Description
» send_notification	boolean	false	none	Determines whether to enable the webhook send notification.
» low_webhook_ids	[integer]	false	none	Webhook IDs for which low alerts are configured
» med_webhook_ids	[integer]	false	none	Webhook IDs for which medium alerts are configured
» high_webhook_ids	[integer]	false	none	Webhook IDs for which high alerts are configured
» cleared_webhook_ids	[integer]	false	none	Webhook IDs for which cleared alerts are configured

Enumerated Values

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%)

Note

This operation does not require authentication

post_*alert_config_server*{server_id}*database*{database_name}*schema*{schema_name}*function*{function_name}*a*

Code samples

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v16/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
}
```

(continues on next page)

(continued from previous page)

```
r = requests.post(https://PEM-SERVER-IP/pem/api/v16/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,
  "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/v16/alert/config/server/{server_id}/database/
↪{database_name}/schema/{schema_name}/function/{function_name}/args/{function_
↪arguments}/,
```

(continues on next page)

(continued from previous page)

```
{
  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}/`

Creates `Alert` for function level.

Body parameter

```
{
  "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": [],
```

(continues on next page)

(continued from previous page)

```
  "high_webhook_ids": [],
  "cleared_webhook_ids": []
}
```

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
database_name	path	string	true	Database name
schema_name	path	string	true	Schema name
function_name	path	string	true	Function name
function_arguments	path	string	true	Comma-separated list of function arguments.
X-Auth-Token	header	string	true	Token of authorization
body	body	*AlertPayload*	true	none

Detailed descriptions

function_arguments: Comma-separated list of function arguments.

Note: Provide space if no function argument

Example responses

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Status 200	None

Response Schema

Note

This operation does not require authentication

get_*alert_config_server*{server_id}*database*{database_name}*schema*{schema_name}*function*{function_name}*arg*

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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 = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v16/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/v16/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}`

Gets `Alerts` for function level by server_id, database_name, schema_name, function_name, and alert_id.

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
database_name	path	string	true	Database name
schema_name	path	string	true	Schema name
function_name	path	string	true	Function name
function_arguments	path	string	true	Comma-separated list of function arguments.
alert_id	path	integer	true	Alert ID
X-Auth-Token	header	string	true	Token of authorization

Detailed descriptions

function_arguments: Comma-separated list of function arguments.

Note: Provide space if no function argument

Example responses


```
{
  "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
```


(continues on next page)

(continued from previous page)

```
  ],
  "med_webhook_ids": [
    0
  ],
  "high_webhook_ids": [
    0
  ],
  "cleared_webhook_ids": [
    0
  ]
}
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	*About alert threads*

Note

This operation does not require authentication

put_*alert_config_server*{server_id}*database*{database_name}*schema*{schema_name}*function*{function_name}*arg*

Code samples

```
#  You can also use wget

curl -X PUT https://PEM-SERVER-IP/pem/api/v16/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
headers = {
  Content-Type: application/json,
  X-Auth-Token: string
}

r = requests.put(https://PEM-SERVER-IP/pem/api/v16/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,
  "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/v16/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();
```

(continues on next page)

(continued from previous page)

```
}).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}
```

Updates `Alert` for function level.

Body parameter

```
{
  "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": []
}
```

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
database_name	path	string	true	Database name
schema_name	path	string	true	Schema name
function_name	path	string	true	Function name
function_arguments	path	string	true	Comma-separated list of function arguments.
alert_id	path	integer	true	Alert ID
X-Auth-Token	header	string	true	Token of authorization
body	body	*AlertPayload*	true	none

Detailed descriptions

function_arguments: Comma-separated list of function arguments.

Note: Provide space if no function argument

Example responses

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Status 200	None

Response Schema

Note

This operation does not require authentication

delete_*alert_config_server*{server_id}*database*{database_name}*schema*{schema_name}*function*{function_name}

Code samples

```
#  You can also use wget

curl -X DELETE https://PEM-SERVER-IP/pem/api/v16/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
}
```

(continues on next page)

(continued from previous page)

```
r = requests.delete(https://PEM-SERVER-IP/pem/api/v16/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/v16/alert/config/server/{server_id}/database/
↪{database_name}/schema/{schema_name}/function/{function_name}/args/{function_
↪arguments}/{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}/function/{function_name}/args/{function_arguments}/{alert_id}`

Deletes the `Alert` object.

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
database_name	path	string	true	Database name
schema_name	path	string	true	Schema name
function_name	path	string	true	Function name
function_arguments	path	string	true	Comma-separated list of function arguments.
alert_id	path	integer	true	Alert ID
X-Auth-Token	header	string	true	Token of authorization

Detailed descriptions

function_arguments: Comma-separated list of function arguments.

Note: Provide space if no function argument

Example responses

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful deletion	None

Response Schema

> **Note**
>
> This operation does not require authentication

Copy Alert

post_*alert_copy_agent*{agent_id}

Code samples

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v16/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/v16/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
};

fetch(https://PEM-SERVER-IP/pem/api/v16/alert/copy/agent/{agent_id},
{
```

(continues on next page)

(continued from previous page)

```
  method: POST,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

```
POST /alert/copy/agent/{agent_id}
```

Copies agent level alert to another agent.

Body parameter

```
[
  {
    "type": "string",
    "agent_id": 0,
    "existing_alert_options": "I"
  }
]
```

Parameters

Name	In	Type	Required	Description
agent_id	path	integer	true	Agent ID
X-Auth-Token	header	string	true	Token of authorization
body	body	*CopyAlertAgentData*	true	none

Detailed descriptions

body: NOTE:

- **type** value must be 'agent'

Example responses

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Status 200	None

Response Schema

> **Note**
>
> This operation does not require authentication

post_*alert_copy_server*{server_id}

Code samples

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v16/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/v16/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,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v16/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}`

Copies server-level alert to another server.

Body parameter

```
[
  {
    "type": "string",
    "server_id": 0,
    "existing_alert_options": "I"
  }
]
```

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
X-Auth-Token	header	string	true	Token of authorization
body	body	*CopyAlert-ServerData*	true	none

Detailed descriptions

body: NOTE:

- **type** value must be 'server'

Example responses

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Status 200	None

Response Schema

> **Note**
>
> This operation does not require authentication

post_*alert_copy_server*{server_id}*database*{database_name}

Code samples

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v16/alert/copy/server/{server_id}/database/
```

(continues on next page)

(continued from previous page)

```
↪{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/v16/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"
  }
];
const headers = {
  Content-Type:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v16/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}`

Copies database-level alert to another database.

Body parameter

```
[
  {
    "type": "string",
    "server_id": 0,
    "database_name": "string",
    "existing_alert_options": "I"
```

(continues on next page)

(continued from previous page)

```
  }
]
```

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
database_name	path	string	true	Database name
X-Auth-Token	header	string	true	Token of authorization
body	body	*CopyAlert-Database-Data*	true	none

Detailed descriptions

body: NOTE:

- **type** value must be 'database'

Example responses

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Status 200	None

Response Schema

Note

This operation does not require authentication

post_*alert_copy_server*{server_id}*database*{database_name}*schema*{schema_name}

Code samples

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v16/alert/copy/server/{server_id}/database/
↪{database_name}/schema/{schema_name} \
  -H Content-Type: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
```

(continues on next page)

(continued from previous page)

```
  Content-Type: application/json,
  X-Auth-Token: string
}

r = requests.post(https://PEM-SERVER-IP/pem/api/v16/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/v16/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}`

Copies schema-level alert to another schema.

Body parameter

```
[
  {
    "type": "string",
    "server_id": 0,
    "database_name": "string",
    "schema_name": "string",
    "existing_alert_options": "I"
  }
]
```

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
database_name	path	string	true	Database name
schema_name	path	string	true	Schema name
X-Auth-Token	header	string	true	Token of authorization
body	body	*Copy-AlertSchema-Data*	true	none

Detailed descriptions

body: NOTE:

- **type** value must be 'schema'

Example responses

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Status 200	None

Response Schema

Note

This operation does not require authentication

post_*alert_copy_server*{server_id}*database*{database_name}*schema*{schema_name}*table*{table_name}

Code samples

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v16/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
}
```

(continues on next page)

(continued from previous page)

```
r = requests.post(https://PEM-SERVER-IP/pem/api/v16/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/v16/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}`

Copies table-level alert to another table.

Body parameter

```
[
  {
    "type": "string",
    "server_id": 0,
    "database_name": "string",
    "schema_name": "string",
    "table_name": "string",
    "existing_alert_options": "I"
  }
]
```

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
database_name	path	string	true	Database name
schema_name	path	string	true	Schema name
table_name	path	string	true	Table name
X-Auth-Token	header	string	true	Token of authorization
body	body	*CopyAlert-TableData*	true	none

Detailed descriptions

body: NOTE:

- **type** value must be 'table'

Example responses

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Status 200	None

Response Schema

> **Note**
>
> This operation does not require authentication

post_*alert_copy_server*{server_id}*database*{database_name}*schema*{schema_name}*index*{index_name}

Code samples

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v16/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
}
```

(continues on next page)

(continued from previous page)

```
r = requests.post(https://PEM-SERVER-IP/pem/api/v16/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/v16/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();
}).then(function(body) {
    console.log(body);
});
```

`POST /alert/copy/server/{server_id}/database/{database_name}/schema/{schema_name}/index/{index_name}`

Copies index-level alert to another index.

Body parameter

```
[
  {
    "type": "string",
    "server_id": 0,
    "database_name": "string",
    "schema_name": "string",
    "index_name": "string",
    "existing_alert_options": "I"
  }
]
```

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
database_name	path	string	true	Database name
schema_name	path	string	true	Schema name
index_name	path	string	true	Index name
X-Auth-Token	header	string	true	Token of authorization
body	body	*CopyAlertIndexData*	true	none

Detailed descriptions

body: NOTE:

- **type** value must be 'index'

Example responses

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Status 200	None

Response Schema

Note

This operation does not require authentication

post_*alert_copy_server*{server_id}*database*{database_name}*schema*{schema_name}*function*{function_name}*arg*

Code samples

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v16/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
}
```

(continues on next page)

(continued from previous page)

```
r = requests.post(https://PEM-SERVER-IP/pem/api/v16/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",
    "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/v16/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}`

Copies function-level alert to another function.

Body parameter

```
[
  {
    "type": "string",
    "server_id": 0,
    "database_name": "string",
    "schema_name": "string",
    "function_name": "string",
    "function_arguments": "string",
    "existing_alert_options": "I"
```

(continues on next page)

(continued from previous page)

```
  }
]
```

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
database_name	path	string	true	Database name
schema_name	path	string	true	Schema name
function_name	path	string	true	Function name
function_arguments	path	string	true	Comma-separated list of function arguments.
X-Auth-Token	header	string	true	Token of authorization
body	body	*CopyAlert-Function-Data*	true	none

Detailed descriptions

function_arguments: Comma-separated list of function arguments.

Note: Provide space if no function argument

body: NOTE:

- **type** value must be 'function'

Example responses

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Status 200	None

Response Schema

Note

This operation does not require authentication

post_*alert_copy_server*{server_id}*database*{database_name}*schema*{schema_name}*sequence*{sequence_name}

Code samples

```
#  You can also use wget
```

(continues on next page)

(continued from previous page)

```
curl -X POST https://PEM-SERVER-IP/pem/api/v16/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/v16/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/v16/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);
});
```

`POST /alert/copy/server/{server_id}/database/{database_name}/schema/{schema_name}/sequence/{sequence_name}`

Copies sequence-level alert to another sequence.

Body parameter

```
[
  {
    "type": "string",
    "server_id": 0,
    "database_name": "string",
    "schema_name": "string",
    "sequence_name": "string",
    "existing_alert_options": "I"
  }
]
```

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
database_name	path	string	true	Database name
schema_name	path	string	true	Schema name
sequence_name	path	string	true	Sequence name
X-Auth-Token	header	string	true	Token of authorization
body	body	*CopyAlertSequenceData*	true	none

Detailed descriptions

body: NOTE:

- **type** value must be 'sequence'

Example responses

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Status 200	None

Response Schema

> **Note**
>
> This operation does not require authentication

Alert template

get_*alert_template*

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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/v16/alert/template/, headers = headers)

print(r.json())
```

```
const headers = {
  Accept:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v16/alert/template/,
{
  method: GET,

  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

`GET /alert/template/`

Gets Alert templates.

Parameters

Name	In	Type	Required	Description
X-Auth-Token	header	string	true	Token of authorization

Example responses

```
[
  {
    "name": "string",
    "description": "string",
    "object_type": 0,
    "param_names": [
      "string"
```

(continues on next page)

(continued from previous page)


```
    ],
    "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"
  }
]
```


Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	Inline

Response Schema

Status Code **200**

Name	Type	Required	Restrictions	Description
anonymous	*Query information*]	false	none	none
» name	string	false	none	Name of the alert template. NOTE: This property is a mandatory parameter for a POST request.
» description	string	false	none	Description of the alert template. NOTE: This property is a mandatory parameter for a POST request.

continues on next page

Table 273 – continued from previous page

Name	Type	Required	Restrictions	Description
» object_type	integer	false	none	Defines the type of the object to which this alert template applies. - A system-wide alert template is 50. - An alert template applicable to an agent is 100. - An alert template applicable to a managed server is 200. - An alert template applicable to a database for a particular monitored server is 300. - An alert template applicable to a schema on a particular database for the managed server is 400. - An alert template applicable to a table on a particular schema for the database on a monitored server is 500. - An alert template applicable to an index on a particular schema for the database on a monitored server is 600. - An alert template applicable to a sequence on a particular schema for the database on a monitored server is 700. - An alert template applicable to a function on a particular schema for the database on a monitored server is 800. NOTE: This property is a mandatory parameter for a POST request.
» param_names	[string]	false	none	Defines the name of parameters.

continues on next page

Table 273 – continued from previous page

Name	Type	Required	Restrictions	Description
» param_types	string	false	none	Defines the type of parameters. Example: INTEGER/STRING..
» params_units	[string]	false	none	Defines the unit used for parameters.Example: 'DAYS','True or False','Hours','Minutes' etc
» sql	string	false	none	SQL query, possibly parameterized, to get a numeric value indicating threshold reached. NOTE: This property is a mandatory parameter for a POST request.
» applicable_on_server	string	false	none	Specifies the type of monitored server to which the template applies. Possible values are ALL, POSTGRES_SERVER, and ADVANCED_SERVER. NOTE: This property is a mandatory parameter for a POST request.
» default_check_frequency	integer	false	none	Defines how frequently (in minutes) to check for the alert defined in this template. NOTE: This property is a mandatory parameter for a POST request.
» default_history_retention	integer	false	none	Defines the history retention period in days for this type of alerts. NOTE: This property is a mandatory parameter for a POST request.
» probe_dependency_list	[string]	false	none	Defines the list of probes on which alert template is dependent.
» threshold_unit	string	false	none	Units used to calculate the threshold, such as MB/kB/seconds/days/... This value is only used for the GUI.

continues on next page

Table 273 – continued from previous page

Name	Type	Required	Restrictions	Description
» is_system_template	boolean	false	none	Determines whether the given template is system-defined or user-defined. NOTE: This property is not applicable for POST/PUT requests.
» is_auto_create	boolean	false	none	Determines whether an alert should be created as soon as an agent or server is added to PEM.Note: Only supported for Agent and Server level i.e object_type 100 and 200 respectively.
» operator	string	false	none	Operator to compare threshold values.Only supported for Agent and Server level i.e object_type 100 and 200 respectively.
» low_threshold_value	number	false	none	Low threshold value.Only supported for Agent and Server level i.e object_type 100 and 200 respectively.
» medium_threshold_value	number	false	none	Medium threshold value.Only supported for Agent and Server level i.e object_type 100 and 200 respectively.
» high_threshold_value	number	false	none	High threshold value.Only supported for Agent and Server level i.e object_type 100 and 200 respectively.
» info_sql	string	false	none	Specifies the SQL query that provides detailed information about the alert on the dashboard.

Enumerated Values

Property	Value
applicable_on_server	ALL
applicable_on_server	POSTGRES_SERVER
applicable_on_server	ADVANCED_SERVER
operator	>
operator	<

> **Note**
>
> This operation does not require authentication

post_*alert_template*

Code samples

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v16/alert/template/ \
  -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/v16/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": ">",
```

(continues on next page)

(continued from previous page)

```
  "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/v16/alert/template/,
{
  method: POST,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

`POST /alert/template/`

Creates alert template.

Body parameter

```
{
  "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,
```

(continues on next page)

(continued from previous page)

```
  "medium_threshold_value": 0,
  "high_threshold_value": 0,
  "info_sql": "string"
}
```

Parameters

Name	In	Type	Required	Description
X-Auth-Token	header	string	true	Token of authorization
body	body	*AlertTemplatePost*	true	none

Example responses

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Status 200	None

Response Schema

Note

This operation does not require authentication

get_*alert_template*{id}

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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/v16/alert/template/{id}, headers =␣
↪headers)

print(r.json())
```

```
const headers = {
  Accept:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v16/alert/template/{id},
{
  method: GET,

  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

`GET /alert/template/{id}`

Gets alert template identified by the ID.

Parameters

Name	In	Type	Required	Description
id	path	integer	true	Alert template ID
X-Auth-Token	header	string	true	Token of authorization

Example responses

```
{
  "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": ">",
```

(continues on next page)

(continued from previous page)

```
  "low_threshold_value": 0,
  "medium_threshold_value": 0,
  "high_threshold_value": 0,
  "info_sql": "string"
}
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	*AlertTemplateParams*

Note

This operation does not require authentication

put_*alert_template*{id}

Code samples

```
#  You can also use wget

curl -X PUT https://PEM-SERVER-IP/pem/api/v16/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/v16/alert/template/{id}, headers =␣
↪headers)

print(r.json())
```

```
const inputBody = {
  "name": "string",
  "description": "string",
  "sql": "string",
  "applicable_on_server": "ALL",
  "default_check_frequency": 0,
  "default_history_retention": 0,
  "probe_dependency_list": {
    "deleted": [
      {
```

(continues on next page)

(continued from previous page)

```
        "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/v16/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}`

Updates alert template.

Body parameter

```
{
  "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"
      }
```

(continues on next page)

(continued from previous page)

```
    ]
  },
  "threshold_unit": "string",
  "is_system_template": false,
  "info_sql": "string"
}
```

Parameters

Name	In	Type	Required	Description
id	path	integer	true	Alert template ID
X-Auth-Token	header	string	true	Token of authorization
body	body	*AlertTemplatePut*	true	none

Example responses

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Status 200	None

Response Schema

Note

This operation does not require authentication

delete_*alert_template*{id}

Code samples

```
#  You can also use wget

curl -X DELETE https://PEM-SERVER-IP/pem/api/v16/alert/template/{id} \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  X-Auth-Token: string
}

r = requests.delete(https://PEM-SERVER-IP/pem/api/v16/alert/template/{id}, headers =␣
↪headers)
```

(continues on next page)

(continued from previous page)

```
print(r.json())
```

```
const headers = {
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v16/alert/template/{id},
{
  method: DELETE,

  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

`DELETE /alert/template/{id}`

Deletes the alert template.

Parameters

Name	In	Type	Required	Description
id	path	integer	true	Alert template ID
X-Auth-Token	header	string	true	Token of authorization

Example responses

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful deletion	None

Response Schema

Note

This operation does not require authentication

Export Custom Alert Templates

post_alert_custom_export

Code samples

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v16/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/v16/alert/custom/export/, headers =␣
↪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/v16/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/`

Exports custom alert templates.

Body parameter

```
{
  "alert_templates": [
```

(continues on next page)

(continued from previous page)

```
    0
  ]
}
```

Parameters

Name	In	Type	Required	Description
X-Auth-Token	header	string	true	Token of authorization
body	body	*CustomAlertTemplateExportData*	true	none

Example responses

```
{
  "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,
```

(continues on next page)

(continued from previous page)


```
                "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"
              }
            ]
          }
        ]
      }
    ]
}
```


Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	Inline

Response Schema

Status Code **200**

Name	Type	Required	Restrictions	Description
» version	integer	false	none	Export version id
» items	*Query information*]	false	none	none
»» version	integer	false	none	Version of the alert template export

continues on next page

Table 285 – continued from previous page

Name	Type	Required	Restrictions	Description
»» alert_templates	*Query information*]	false	none	none
»»» name	string	false	none	Name of the alert template. NOTE: This property is a mandatory parameter for a POST request.
»»» description	string	false	none	Description of the alert template. NOTE: This property is a mandatory parameter for a POST request.
»»» reference_id	string	false	none	Internal identifier for the alert template
»»» default_history_retention	integer	false	none	Defines the history retention period in days for this type of alerts. NOTE: This property is a mandatory parameter for a POST request.

continues on next page

Table 285 – continued from previous page

Name	Type	Required	Restrictions	Description
»»» object_type	integer	false	none	Defines the type of the object to which this alert template applies. - A system-wide alert template is 50. - An alert template applicable to an agent is 100. - An alert template applicable to a managed server is 200. - An alert template applicable to a database for a particular monitored server is 300. - An alert template applicable to a schema on a particular database for the managed server is 400. - An alert template applicable to a table on a particular schema for the database on a monitored server is 500. - An alert template applicable to an index on a particular schema for the database on a monitored server is 600. - An alert template applicable to a sequence on a particular schema for the database on a monitored server is 700. - An alert template applicable to a function on a particular schema for the database on a monitored server is 800. NOTE: This property is a mandatory parameter for a POST request.

continues on next page

Table 285 – continued from previous page

Name	Type	Required	Restrictions	Description
»»»» sql	string	false	none	SQL query, possibly parameterized, to get a numeric value indicating the threshold reached. NOTE: This property is a mandatory parameter for a POST request.
»»»» applicable_on_server	string	false	none	Specifies the type of monitored server to which the template applies. Possible values are ALL, POSTGRES_SERVER, ADVANCED_SERVER. NOTE: This property is a mandatory parameter for a POST request.
»»»» default_check_frequency	integer	false	none	Specifies how frequently (in minutes) to check for the alert that this template defines. NOTE: This property is a mandatory parameter for a POST request.
»»»» threshold_unit	string	false	none	Units used to calculate the threshold, such as MB/kB/seconds/days/... This value is only used for the GUI.
»»»» is_system_template	boolean	false	none	Determines whether the given template is system-defined or user-defined. NOTE: This property is not applicable for POST/PUT requests.
»»»» is_auto_create	boolean	false	none	Determines whether an alert should be created as soon as an agent or server is added to PEM..
»»»» thresholds	[number]	false	none	Default threshold values. Values in the thresholds array must be in a sorted order based on the operator.
»»»» operator	string	false	none	Operator to compare threshold values
»»»» high_threshold_value	string	false	none	High threshold value
»»»» medium_threshold_value	string	false	none	Medium threshold value
»»»» low_threshold_value	string	false	none	Low threshold value

continues on next page

Table 285 – continued from previous page

Name	Type	Required	Restrictions	Description
»»» info_sql	string	false	none	Defines the SQL query that provides detailed information about the alert on the dashboard.
»»» probes	[allOf]	false	none	none

- allOf*

Name	Type	Required	Restrictions	Description
»»»» *anonymous*	*CustomProbePostData*	false	none	none
»»»»» probe_name	string	false	none	Name of the probe
»»»»» collection_method	string	false	none	Defines the method of statistics collection. PEM supports three methods for custom made probes: SQL, WMI, or batch/shell script. 'w' stands for WMI script. This type of script only applies to the Windows platform, and the target_type is 100 (for agent). 'b' stands for batch/shell script. The 'target_type' for this script is always 100 (agent). This script behaves as a batch script on Windows and a shell script on other platforms. 's' stands for SQL script. The 'target_type' is always 200 (server), 300 (database), 400 (schema), 500 (table), 600 (index), 700 (sequence), 800 (function), 900 (view) or 1000 (extension).
»»»»» enabled	boolean	false	none	Determines whether whether the probe is enabled/disabled by default.

continues on next page

Table 286 – continued from previous page

Name	Type	Required	Restrictions	Description
»»»»»»» target_type	integer	false	none	Defines the type of the object to which the probe applies. - A probe with target_type 100 is responsible for collecting statistics for an agent. - A probe with target_type 200 is responsible for collecting statistics for a monitored server. - A probe with target_type 300 is responsible for collecting statistics for a database. - A probe with target_type 400 is responsible for collecting statistics for a schema. - A probe with target_type 500 is responsible for collecting statistics for a table. - A probe with target_type 600 is responsible for collecting statistics for a index. - A probe with target_type 700 is responsible for collecting statistics for a sequence. - A probe with target_type 800 is responsible for collecting statistics for a function. - A probe with target_type 900 is responsible for collecting statistics for a view. - A probe with target_type 1000 is responsible for collecting statistics for a extension.
»»»»»»» any_server_version	boolean	false	none	Determines whether this probe applies to any version of a Postgres variant. For more details, please consult the documentation.

continues on next page

Table 286 – continued from previous page

Name	Type	Required	Restrictions	Description
»»»»»»» any_extension_version	boolean	false	none	Determines whether this probe applies to any version of a Extension variant. For more details, please consult the documentation.
»»»»»»» extension_name	string	false	none	Name of the extension. Valid names are edbspl , plpgsql , hstore , pldbgapi , edb_dblink_oci , edb_dblink_libpq , sslutils .
»»»»»»» interval	integer	false	none	Defines how frequently the probe collects statistics (in seconds).Must be greater than or equal to 10
»»»»»»» lifetime	integer	false	none	Defines the history data retention time (in days). Valid range is 1-365. This value does not have any effect if 'discard_history' is set to true.
»»»»»»» discard_history	boolean	false	none	Defines whether to discard history data. When 'discard_history' is set to false, no history data is stored for this probe.
»»»»»»» platform	string	false	none	Defines the platform where this probe can run. This value only applies to an agent-level probe.
»»»»»»» probe_code	string	false	none	A SQL/Batch script/Shell script/WMI script to collect the statistics as per definition of the code.
»»»»»»» probe_columns	[object]	false	none	none
»»»»»»»» pc_name	string	false	none	Column name

continues on next page

Table 286 – continued from previous page

Name	Type	Required	Restrictions	Description
»»»»»»»» pc_data_type	string	false	none	Data type of the column. Supported datatypes are: bigint char decimal double precision integer numeric real text tex tamp timestamptz
»»»»»»»» pc_unit	string	false	none	Returns the unit of the column.
»»»»»»»» pc_col_type	string	false	none	Returns the columns as part of the primary key of the internal table, or not. 'k' stands for a key column, and 'm' stands for a member/normal column.
»»»»»»»» pc_graphable	boolean	false	none	Determines whether the data of the column can be graphed on a chart.
»»»»»»»» pc_calc_pit	boolean	false	none	Determines whether the server calculates point-in-time for metric data for this column.
»»»»»»»» pc_pit_default	boolean	false	none	Determines whether metric data is stored by point-in-time by default. Please refer to the 'Custom Probes' section in the documentation for detailed information.
»»»»»»» alternate_code	*Query information*]	false	none	none
»»»»»»»» server_version_id	integer	false	none	Server version ID for the monitored server
»»»»»»»» server_probe_code	string	false	none	SQL query to run on the specified server version
»»»»»»»» extension_version	integer	false	none	Specifies the extension version. Valid for extension target type

- and*

Name	Type	Required	Restrictions	Description
»»»»» *anonymous*	object	false	none	none
»»»»»»» internal_name	string	false	none	Internal name of the probe from the system where probe was exported

Enumerated Values

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
pc_col_type	k

Note

This operation does not require authentication

Import Custom Alert Templates

post_*alert_custom_import*

Code samples

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v16/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,
```

(continues on next page)

(continued from previous page)

```
  X-Auth-Token: string
}

r = requests.post(https://PEM-SERVER-IP/pem/api/v16/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,
      "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",
```

(continues on next page)

(continued from previous page)

```
              "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/v16/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/`

Imports custom alert templates using exported alert templates.

Body parameter

```
{
  "version": 1,
  "skip_overwrite": true,
  "skip_overwrite_probe": true,
  "alert_templates": [
    {
      "name": "string",
      "description": "string",
      "reference_id": "string",
      "default_history_retention": 0,
```

(continues on next page)

(continued from previous page)


```
      "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,
              "pc_calc_pit": true,
              "pc_pit_default": true
            }
          ],
          "alternate_code": [
            {
              "server_version_id": 0,
              "server_probe_code": "string",
              "extension_version": 0
            }
          ],
          "internal_name": "string"
        }
      ]
    }
  ]
```


(continues on next page)

(continued from previous page)

```
}
```

Parameters

Name	In	Type	Required	Description
X-Auth-Token	header	string	true	Token of authorization
body	body	*CustomAlertTemplateImportPostData*	true	none

Example responses

```
[
  {
    "name": "string",
    "status": "string",
    "msg": "string"
  }
]
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Status 200	Inline

Response Schema

Status Code **200**

Name	Type	Required	Restrictions	Description
» name	string	true	none	none
» status	string	true	none	none
» msg	string	true	none	none

Note

This operation does not require authentication

Export Custom Charts

post_*chart_custom_export*

Code samples

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v16/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/v16/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/v16/chart/custom/export/,
{
  method: POST,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

`POST /chart/custom/export/`

Exports custom charts.

Body parameter

```
{
  "charts": [
    0
  ]
}
```

Parameters

Name	In	Type	Required	Description
X-Auth-Token	header	string	true	Token of authorization
body	body	*CustomCharExportData*	true	none

Example responses


```
{
  "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,
              "platform": "unix",
              "probe_code": "string",
              "probe_columns": [
                {
                  "pc_name": "string",
```


(continues on next page)

(continued from previous page)


```
                    "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"
              }
            ]
          }
        ]
      }
    ]
}
```


Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	Inline

Response Schema

Status Code **200**

Name	Type	Required	Restrictions	Description
» version	integer	false	none	Export version id
» items	*Query information*]	false	none	none
»» version	integer	false	none	Version of the alert template export
»» alert_templates	*Query information*]	false	none	none
»»» name	string	false	none	Name of the alert template. NOTE: This property is a mandatory parameter for a POST request.

continues on next page

Table 294 – continued from previous page

Name	Type	Required	Restrictions	Description
»»»» description	string	false	none	Description of the alert template. NOTE: This property is a mandatory parameter for a POST request.
»»»» reference_id	string	false	none	Internal identifier for the alert template
»»»» default_history_retention	integer	false	none	Defines the history retention period in days for this type of alerts. NOTE: This property is a mandatory parameter for a POST request.

continues on next page

Table 294 – continued from previous page

Name	Type	Required	Restrictions	Description
»»» object_type	integer	false	none	Defines the type of the object to which this alert template applies. - A system-wide alert template is 50. - An alert template applicable to an agent is 100. - An alert template applicable to a managed server is 200. - An alert template applicable to a database for a particular monitored server is 300. - An alert template applicable to a schema on a particular database for the managed server is 400. - An alert template applicable to a table on a particular schema for the database on a monitored server is 500. - An alert template applicable to an index on a particular schema for the database on a monitored server is 600. - An alert template applicable to a sequence on a particular schema for the database on a monitored server is 700. - An alert template applicable to a function on a particular schema for the database on a monitored server is 800. NOTE: This property is a mandatory parameter for a POST request.

continues on next page

Table 294 – continued from previous page

Name	Type	Required	Restrictions	Description
»»»» sql	string	false	none	SQL query, possibly parameterized, to get a numeric value indicating the threshold reached. NOTE: This property is a mandatory parameter for a POST request.
»»»» applicable_on_server	string	false	none	Specifies the type of monitored server to which the template applies. Possible values are ALL, POSTGRES_SERVER, ADVANCED_SERVER. NOTE: This property is a mandatory parameter for a POST request.
»»»» default_check_frequency	integer	false	none	Specifies how frequently (in minutes) to check for the alert that this template defines. NOTE: This property is a mandatory parameter for a POST request.
»»»» threshold_unit	string	false	none	Units used to calculate the threshold, such as MB/kB/seconds/days/... This value is only used for the GUI.
»»»» is_system_template	boolean	false	none	Determines whether the given template is system-defined or user-defined. NOTE: This property is not applicable for POST/PUT requests.
»»»» is_auto_create	boolean	false	none	Determines whether an alert should be created as soon as an agent or server is added to PEM..
»»»» thresholds	[number]	false	none	Default threshold values. Values in the thresholds array must be in a sorted order based on the operator.
»»»» operator	string	false	none	Operator to compare threshold values
»»»» high_threshold_value	string	false	none	High threshold value
»»»» medium_threshold_value	string	false	none	Medium threshold value
»»»» low_threshold_value	string	false	none	Low threshold value

continues on next page

Table 294 – continued from previous page

Name	Type	Required	Restrictions	Description
»»» info_sql	string	false	none	Defines the SQL query that provides detailed information about the alert on the dashboard.
»»» probes	[allOf]	false	none	none

- allOf*

Name	Type	Required	Restrictions	Description
»»» *anonymous*	*CustomProbePostData*	false	none	none
»»»» probe_name	string	false	none	Name of the probe
»»»» collection_method	string	false	none	Defines the method of statistics collection. PEM supports three methods for custom made probes: SQL, WMI, or batch/shell script. 'w' stands for WMI script. This type of script only applies to the Windows platform, and the target_type is 100 (for agent). 'b' stands for batch/shell script. The 'target_type' for this script is always 100 (agent). This script behaves as a batch script on Windows and a shell script on other platforms. 's' stands for SQL script. The 'target_type' is always 200 (server), 300 (database), 400 (schema), 500 (table), 600 (index), 700 (sequence), 800 (function), 900 (view) or 1000 (extension).
»»»» enabled	boolean	false	none	Determines whether whether the probe is enabled/disabled by default.

continues on next page

Table 295 – continued from previous page

Name	Type	Required	Restrictions	Description
»»»»»»» target_type	integer	false	none	Defines the type of the object to which the probe applies. - A probe with target_type 100 is responsible for collecting statistics for an agent. - A probe with target_type 200 is responsible for collecting statistics for a monitored server. - A probe with target_type 300 is responsible for collecting statistics for a database. - A probe with target_type 400 is responsible for collecting statistics for a schema. - A probe with target_type 500 is responsible for collecting statistics for a table. - A probe with target_type 600 is responsible for collecting statistics for a index. - A probe with target_type 700 is responsible for collecting statistics for a sequence. - A probe with target_type 800 is responsible for collecting statistics for a function. - A probe with target_type 900 is responsible for collecting statistics for a view. - A probe with target_type 1000 is responsible for collecting statistics for a extension.
»»»»»»» any_server_version	boolean	false	none	Determines whether this probe applies to any version of a Postgres variant. For more details, please consult the documentation.

continues on next page

Table 295 – continued from previous page

Name	Type	Required	Restrictions	Description
»»»»»»» any_extension_version	boolean	false	none	Determines whether this probe applies to any version of a Extension variant. For more details, please consult the documentation.
»»»»»»» extension_name	string	false	none	Name of the extension. Valid names are edbspl , plpgsql , hstore , pldbgapi , edb_dblink_oci , edb_dblink_libpq , sslutils .
»»»»»»» interval	integer	false	none	Defines how frequently the probe collects statistics (in seconds).Must be greater than or equal to 10
»»»»»»» lifetime	integer	false	none	Defines the history data retention time (in days). Valid range is 1-365. This value does not have any effect if 'discard_history' is set to true.
»»»»»»» discard_history	boolean	false	none	Defines whether to discard history data. When 'discard_history' is set to false, no history data is stored for this probe.
»»»»»»» platform	string	false	none	Defines the platform where this probe can run. This value only applies to an agent-level probe.
»»»»»»» probe_code	string	false	none	A SQL/Batch script/Shell script/WMI script to collect the statistics as per definition of the code.
»»»»»»» probe_columns	[object]	false	none	none
»»»»»»»» pc_name	string	false	none	Column name

continues on next page

Table 295 – continued from previous page

Name	Type	Required	Restrictions	Description
»»»»»»»» pc_data_type	string	false	none	Data type of the column. Supported datatypes are: bigint char decimal double precision integer numeric real text tex tamp timestamptz
»»»»»»»» pc_unit	string	false	none	Returns the unit of the column.
»»»»»»»» pc_col_type	string	false	none	Returns the columns as part of the primary key of the internal table, or not. 'k' stands for a key column, and 'm' stands for a member/normal column.
»»»»»»»» pc_graphable	boolean	false	none	Determines whether the data of the column can be graphed on a chart.
»»»»»»»» pc_calc_pit	boolean	false	none	Determines whether the server calculates point-in-time for metric data for this column.
»»»»»»»» pc_pit_default	boolean	false	none	Determines whether metric data is stored by point-in-time by default. Please refer to the 'Custom Probes' section in the documentation for detailed information.
»»»»»»» alternate_code	*Query information*]	false	none	none
»»»»»»»» server_version_id	integer	false	none	Server version ID for the monitored server
»»»»»»»» server_probe_code	string	false	none	SQL query to run on the specified server version
»»»»»»»» extension_version	integer	false	none	Specifies the extension version. Valid for extension target type

- and*

Name	Type	Required	Restrictions	Description
»»»»» *anonymous*	object	false	none	none
»»»»»»» internal_name	string	false	none	Internal name of the probe from the system where probe was exported

Enumerated Values

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
pc_col_type	k

Note

This operation does not require authentication

Import Custom Charts

post_*chart_custom_import*

Code samples

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v16/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,
```

(continues on next page)

(continued from previous page)

```
  X-Auth-Token: string
}

r = requests.post(https://PEM-SERVER-IP/pem/api/v16/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",
      "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
    }
  ]
```

(continues on next page)

(continued from previous page)

```
};
const headers = {
  Content-Type:application/json,
  Accept:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v16/chart/custom/import/,
{
  method: POST,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

`POST /chart/custom/import/`

Imports custom alert templates using exported charts.

Body parameter

```
{
  "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,
```

(continues on next page)

(continued from previous page)


```
      "line_span": 0,
      "sel_metrics_L": [
        {}
      ],
      "sel_metrics_T": [
        {}
      ],
      "sel_metrics_C": [
        {}
      ],
      "shared": [
        0
      ],
      "shared_all": true
    }
  ]
}
```


Parameters

Name	In	Type	Required	Description
X-Auth-Token	header	string	true	Token of authorization
body	body	*Custom-ChartImport-PostData*	true	none

Example responses


```
[
  {
    "name": "string",
    "status": "string",
    "msg": "string"
  }
]
```


Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Status 200	Inline

Response Schema

Status Code **200**

Name	Type	Required	Restrictions	Description
» name	string	true	none	none
» status	string	true	none	none
» msg	string	true	none	none

Note

This operation does not require authentication

Export Custom Dashboards

post_*dashboard_custom_export*

Code samples

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v16/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/v16/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/v16/dashboard/custom/export/,
{
  method: POST,
  body: inputBody,
  headers: headers
```

(continues on next page)

(continued from previous page)

```
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

```
POST /dashboard/custom/export/
```

Exports custom dashboards.

Body parameter

```
{
  "dashboards": [
    0
  ]
}
```

Parameters

Name	In	Type	Required	Description
X-Auth-Token	header	string	true	Token of authorization
body	body	*CustomDashboardExportData*	true	none

Example responses

```
{
  "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": ">",
```

(continues on next page)

(continued from previous page)


```
          "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"
            }
          ]
        }
      ]
    }
  ]
}
```


Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	Inline

Response Schema

Status Code **200**

Name	Type	Required	Restrictions	Description
» version	integer	false	none	Export version id
» items	*Query information*]	false	none	none
»» version	integer	false	none	Version of the alert template export
»» alert_templates	*Query information*]	false	none	none
»»» name	string	false	none	Name of the alert template. NOTE: This property is a mandatory parameter for a POST request.
»»» description	string	false	none	Description of the alert template. NOTE: This property is a mandatory parameter for a POST request.
»»» reference_id	string	false	none	Internal identifier for the alert template
»»» default_history_retention	integer	false	none	Defines the history retention period in days for this type of alerts. NOTE: This property is a mandatory parameter for a POST request.

continues on next page

Table 303 – continued from previous page

Name	Type	Required	Restrictions	Description
»»» object_type	integer	false	none	Defines the type of the object to which this alert template applies. - A system-wide alert template is 50. - An alert template applicable to an agent is 100. - An alert template applicable to a managed server is 200. - An alert template applicable to a database for a particular monitored server is 300. - An alert template applicable to a schema on a particular database for the managed server is 400. - An alert template applicable to a table on a particular schema for the database on a monitored server is 500. - An alert template applicable to an index on a particular schema for the database on a monitored server is 600. - An alert template applicable to a sequence on a particular schema for the database on a monitored server is 700. - An alert template applicable to a function on a particular schema for the database on a monitored server is 800. NOTE: This property is a mandatory parameter for a POST request.

continues on next page

Table 303 – continued from previous page

Name	Type	Required	Restrictions	Description
»»»» sql	string	false	none	SQL query, possibly parameterized, to get a numeric value indicating the threshold reached. NOTE: This property is a mandatory parameter for a POST request.
»»»» applicable_on_server	string	false	none	Specifies the type of monitored server to which the template applies. Possible values are ALL, POSTGRES_SERVER, ADVANCED_SERVER. NOTE: This property is a mandatory parameter for a POST request.
»»»» default_check_frequency	integer	false	none	Specifies how frequently (in minutes) to check for the alert that this template defines. NOTE: This property is a mandatory parameter for a POST request.
»»»» threshold_unit	string	false	none	Units used to calculate the threshold, such as MB/kB/seconds/days/... This value is only used for the GUI.
»»»» is_system_template	boolean	false	none	Determines whether the given template is system-defined or user-defined. NOTE: This property is not applicable for POST/PUT requests.
»»»» is_auto_create	boolean	false	none	Determines whether an alert should be created as soon as an agent or server is added to PEM..
»»»» thresholds	[number]	false	none	Default threshold values. Values in the thresholds array must be in a sorted order based on the operator.
»»»» operator	string	false	none	Operator to compare threshold values
»»»» high_threshold_value	string	false	none	High threshold value
»»»» medium_threshold_value	string	false	none	Medium threshold value
»»»» low_threshold_value	string	false	none	Low threshold value

continues on next page

Table 303 – continued from previous page

Name	Type	Required	Restrictions	Description
»»»» info_sql	string	false	none	Defines the SQL query that provides detailed information about the alert on the dashboard.
»»»» probes	[allOf]	false	none	none

- allOf*

Name	Type	Required	Restrictions	Description
»»»»» *anonymous*	*CustomProbePostData*	false	none	none
»»»»»» probe_name	string	false	none	Name of the probe
»»»»»» collection_method	string	false	none	Defines the method of statistics collection. PEM supports three methods for custom made probes: SQL, WMI, or batch/shell script. 'w' stands for WMI script. This type of script only applies to the Windows platform, and the target_type is 100 (for agent). 'b' stands for batch/shell script. The 'target_type' for this script is always 100 (agent). This script behaves as a batch script on Windows and a shell script on other platforms. 's' stands for SQL script. The 'target_type' is always 200 (server), 300 (database), 400 (schema), 500 (table), 600 (index), 700 (sequence), 800 (function), 900 (view) or 1000 (extension).
»»»»»» enabled	boolean	false	none	Determines whether whether the probe is enabled/disabled by default.

continues on next page

Table 304 – continued from previous page

Name	Type	Required	Restrictions	Description
»»»»»»» target_type	integer	false	none	Defines the type of the object to which the probe applies. - A probe with target_type 100 is responsible for collecting statistics for an agent. - A probe with target_type 200 is responsible for collecting statistics for a monitored server. - A probe with target_type 300 is responsible for collecting statistics for a database. - A probe with target_type 400 is responsible for collecting statistics for a schema. - A probe with target_type 500 is responsible for collecting statistics for a table. - A probe with target_type 600 is responsible for collecting statistics for a index. - A probe with target_type 700 is responsible for collecting statistics for a sequence. - A probe with target_type 800 is responsible for collecting statistics for a function. - A probe with target_type 900 is responsible for collecting statistics for a view. - A probe with target_type 1000 is responsible for collecting statistics for a extension.
»»»»»»» any_server_version	boolean	false	none	Determines whether this probe applies to any version of a Postgres variant. For more details, please consult the documentation.

continues on next page

Table 304 – continued from previous page

Name	Type	Required	Restrictions	Description
»»»»»»» any_extension_version	boolean	false	none	Determines whether this probe applies to any version of a Extension variant. For more details, please consult the documentation.
»»»»»»» extension_name	string	false	none	Name of the extension. Valid names are edbspl , plpgsql , hstore , pldbgapi , edb_dblink_oci , edb_dblink_libpq , sslutils .
»»»»»»» interval	integer	false	none	Defines how frequently the probe collects statistics (in seconds).Must be greater than or equal to 10
»»»»»»» lifetime	integer	false	none	Defines the history data retention time (in days). Valid range is 1-365. This value does not have any effect if 'discard_history' is set to true.
»»»»»»» discard_history	boolean	false	none	Defines whether to discard history data. When 'discard_history' is set to false, no history data is stored for this probe.
»»»»»»» platform	string	false	none	Defines the platform where this probe can run. This value only applies to an agent-level probe.
»»»»»»» probe_code	string	false	none	A SQL/Batch script/Shell script/WMI script to collect the statistics as per definition of the code.
»»»»»»» probe_columns	[object]	false	none	none
»»»»»»»» pc_name	string	false	none	Column name

continues on next page

Table 304 – continued from previous page

Name	Type	Required	Restrictions	Description
»»»»»»»» pc_data_type	string	false	none	Data type of the column. Supported datatypes are: bigint char decimal double precision integer numeric real text tex tamp timestamptz
»»»»»»»» pc_unit	string	false	none	Returns the unit of the column.
»»»»»»»» pc_col_type	string	false	none	Returns the columns as part of the primary key of the internal table, or not. 'k' stands for a key column, and 'm' stands for a member/normal column.
»»»»»»»» pc_graphable	boolean	false	none	Determines whether the data of the column can be graphed on a chart.
»»»»»»»» pc_calc_pit	boolean	false	none	Determines whether the server calculates point-in-time for metric data for this column.
»»»»»»»» pc_pit_default	boolean	false	none	Determines whether metric data is stored by point-in-time by default. Please refer to the 'Custom Probes' section in the documentation for detailed information.
»»»»»»» alternate_code	*Query information*]	false	none	none
»»»»»»»» server_version_id	integer	false	none	Server version ID for the monitored server
»»»»»»»» server_probe_code	string	false	none	SQL query to run on the specified server version
»»»»»»»» extension_version	integer	false	none	Specifies the extension version. Valid for extension target type

- and*

Name	Type	Required	Restrictions	Description
»»»»» *anonymous*	object	false	none	none
»»»»»»» internal_name	string	false	none	Internal name of the probe from the system where probe was exported

Enumerated Values

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
pc_col_type	k

Note

This operation does not require authentication

Import Custom Dashboards

post_*dashboard_custom_import*

Code samples

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v16/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,
```

(continues on next page)

(continued from previous page)

```
  X-Auth-Token: string
}

r = requests.post(https://PEM-SERVER-IP/pem/api/v16/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",
      "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/v16/dashboard/custom/import/,
{
  method: POST,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
```

(continues on next page)

(continued from previous page)

```
});
```

`POST /dashboard/custom/import/`

Imports custom alert templates using exported dashboards.

Body parameter

```
{
  "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
    }
  ]
}
```

Parameters

Name	In	Type	Required	Description
X-Auth-Token	header	string	true	Token of authorization
body	body	*CustomDashboardImportPostData*	true	none

Example responses

```
[
  {
    "name": "string",
    "status": "string",
```

(continues on next page)

(continued from previous page)

```
    "msg": "string"
  }
]
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Status 200	Inline

Response Schema

Status Code **200**

Name	Type	Required	Restrictions	Description
» name	string	true	none	none
» status	string	true	none	none
» msg	string	true	none	none

Note

This operation does not require authentication

Custom Probe

get_*probe*

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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/v16/probe/, headers = headers)

print(r.json())
```

```
const headers = {
  Accept:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v16/probe/,
{
  method: GET,

  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

`GET /probe/`

Gets all the custom/system probes.

Parameters

Name	In	Type	Required	Description
X-Auth-Token	header	string	true	Token of authorization

Example responses

```
[
  {
    "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",
```

(continues on next page)

(continued from previous page)


```
        "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
      }
    ]
  }
]
```


Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	Inline

Response Schema

Status Code **200**

Name	Type	Required	Restrictions	Description
anonymous	*Query information*]	false	none	none
» probe_id	integer	false	none	Identifier for a probe
» probe_name	string	false	none	Name of the probe
» enabled	boolean	false	none	Determines whether the probe is enabled or disabled by default.

continues on next page

Table 312 – continued from previous page

Name	Type	Required	Restrictions	Description
» target_type	integer	false	none	Defines the type of the object to which the probe applies. - A probe with target_type 100 is responsible for collecting statistics for an agent. - A probe with target_type 200 is responsible for collecting statistics for a monitored server. - A probe with target_type 300 is responsible for collecting statistics for a database. - A probe with target_type 400 is responsible for collecting statistics for a schema. - A probe with target_type 500 is responsible for collecting statistics for a table. - A probe with target_type 600 is responsible for collecting statistics for a index. - A probe with target_type 700 is responsible for collecting statistics for a sequence. - A probe with target_type 800 is responsible for collecting statistics for a function. - A probe with target_type 900 is responsible for collecting statistics for a view. - A probe with target_type 1000 is responsible for collecting statistics for a extension.

continues on next page

Table 312 – continued from previous page

Name	Type	Required	Restrictions	Description
» internal_name	string	false	none	Defines the internal name of the probe. The PEM system creates internal tables for storing probe statistics in the 'pemdata', and 'pemhistory' schema. The names of those tables are the same as the internal name.
» is_system_probe	boolean	false	none	Determines whether this probe is user-defined. This flag is always false for a user-defined probe.
» any_server_version	boolean	false	none	Determines whether this probe applies to any version of a Postgres variant. For more details, please consult the documentation.

continues on next page

Table 312 – continued from previous page

Name	Type	Required	Restrictions	Description
» collection_method	string	false	none	Defines the method of statistics collection. PEM supports three methods for custom made probes: SQL, WMI, or batch/shell script. 'w' stands for WMI script. This type of script only applies to the Windows platform, and the target_type is 100 (for agent). 'b' stands for batch/shell script. The 'target_type' for this script is always 100 (agent). This script behaves as a batch script on Windows and a shell script on other platforms. 's' stands for SQL script. The 'target_type' is always 200 (server), 300 (database), 400 (schema), 500 (table), 600 (index), 700 (sequence), 800 (function), or 900 (view), or 1000 (extension).
» interval	integer	false	none	Defines how frequently the probe collects statistics (in seconds).Must be greater than or equal to 10
» lifetime	integer	false	none	Defines the history data retention time (in days). Valid range is 1-365. This value does not have any effect if 'discard_history' is set to true.
» discard_history	boolean	false	none	Determines whether to discard history data. When 'discard_history' is set to false, history of the data for this probe is not stored.

continues on next page

Table 312 – continued from previous page

Name	Type	Required	Restrictions	Description
» platform	string	false	none	Defines the platform where this probe can run. This value only applies to an agent-level probe.
» probe_code	string	false	none	A SQL/Batch script/Shell script/WMI script to collect the statistics as per the definition of the code.
» probe_columns	[object]	false	none	none
»» pc_id	integer	false	none	Column identifier
»» pc_name	string	false	none	Column name
»» pc_data_type	string	false	none	Data type of the column. Supported datatypes are: bigint char decimal double precision integer numeric real text tex tamp timestamptz
»» pc_internal_name	string	false	none	Returns the internal name of the column.
»» pc_unit	string	false	none	Returns the unit of the column.
»» pc_col_type	string	false	none	Returns the column as part of the primary key of the internal table, or not. 'k' stands for a key column, and 'm' stands for a member/normal column.
»» pc_position	integer	false	none	Returns the position of the column in the internal table.
»» pc_graphable	boolean	false	none	Determines whether the data of the column can be graphed on a chart.
»» pc_calc_pit	boolean	false	none	Determines whether the server calculates the point-in-time for metric data for this column.
»» pc_pit_default	boolean	false	none	Determines whether the metric is stored by point-in-time by default. Please refer to the 'Custom Probes' section in the documentation for detailed information.

continues on next page

Table 312 – continued from previous page

Name	Type	Required	Restrictions	Description
» alternate_code	*Query information*]	false	none	none
»» server_version_id	integer	false	none	Server version ID for the monitored server
»» server_probe_code	string	false	none	SQL query to run on the specified server version
»» extension_version	integer	false	none	Specifies the extension version. Valid for extension target type

Enumerated Values

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

Note

This operation does not require authentication

post_*probe*

Code samples

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v16/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/v16/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
};

fetch(https://PEM-SERVER-IP/pem/api/v16/probe/,
{
  method: POST,
  body: inputBody,
  headers: headers
```

(continues on next page)

(continued from previous page)

```
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

```
POST /probe/
```

Creates a new custom probe.

Body parameter

```
{
  "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
    }
  ]
}
```

Parameters

Name	In	Type	Required	Description
X-Auth-Token	header	string	true	Token of authorization

continues on next page

Table 314 – continued from previous page

Name	In	Type	Required	Description
body	body	*CustomProbePostData*	true	none

Example responses

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Status 200	None

Response Schema

> **Note**
>
> This operation does not require authentication

get_*probe*{probe_id}

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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/v16/probe/{probe_id}, headers =␣
↪headers)

print(r.json())
```

```
const headers = {
  Accept:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v16/probe/{probe_id},
{
```

(continues on next page)

(continued from previous page)

```
  method: GET,

  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

```
GET /probe/{probe_id}
```

Gets probe information for specified probe ID.

Parameters

Name	In	Type	Required	Description
probe_id	path	integer	true	Probe ID for which probe information will be returned
X-Auth-Token	header	string	true	Token of authorization

Example responses


```
{
  "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
    }
  ],
```


(continues on next page)

(continued from previous page)

```
  "alternate_code": [
    {
      "server_version_id": 0,
      "server_probe_code": "string",
      "extension_version": 0
    }
  ]
}
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	*CustomProbes*

Note

This operation does not require authentication

put_*probe*{probe_id}

Code samples

```
#  You can also use wget

curl -X PUT https://PEM-SERVER-IP/pem/api/v16/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/v16/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": {
```

(continues on next page)

(continued from previous page)

```
    "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
      }
    ]
  }
};
const headers = {
  Content-Type:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v16/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}`

Updates custom/system probe information for specified probe ID.

Body parameter

```
{
  "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",
        "extension_version": 0
      }
    ],
    "deleted": [
      {
        "server_version_id": 0
      }
    ],
    "added": [
      {
        "server_version_id": 0,
        "server_probe_code": "string",
        "extension_version": 0
      }
    ]
  }
}
```


Parameters

Name	In	Type	Required	Description
probe_id	path	integer	true	Probe ID for which probe information will be returned
X-Auth-Token	header	string	true	Token of authorization
body	body	*CustomProbePutData*	true	none

Example responses

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Status 200	None

Response Schema

> **Note**
>
> This operation does not require authentication

delete_*probe*{probe_id}

Code samples

```
#  You can also use wget

curl -X DELETE https://PEM-SERVER-IP/pem/api/v16/probe/{probe_id} \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  X-Auth-Token: string
}

r = requests.delete(https://PEM-SERVER-IP/pem/api/v16/probe/{probe_id}, headers =␣
↪headers)

print(r.json())
```

```
const headers = {
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v16/probe/{probe_id},
{
  method: DELETE,

  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

```
DELETE /probe/{probe_id}
```

Deletes the custom probe.

Parameters

Name	In	Type	Required	Description
probe_id	path	integer	true	Probe ID for which probe information will be returned
X-Auth-Token	header	string	true	Token of authorization

Example responses

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful deletion	None

Response Schema

> **Note**
>
> This operation does not require authentication

Export Custom Probes

post_probe_custom_export

Code samples

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v16/probe/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/v16/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/v16/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/
```

Exports custom probes.

Body parameter

```
{
  "probes": [
    0
  ]
}
```

Parameters

Name	In	Type	Required	Description
X-Auth-Token	header	string	true	Token of authorization
body	body	*CustomProbeExportData*	true	none

Example responses

```
{
  "version": 0,
  "items": [
    {
      "version": 1,
      "alert_templates": [
        {
```

(continues on next page)

(continued from previous page)

```
            "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,
                    "pc_calc_pit": true,
                    "pc_pit_default": true
                  }
                ],
                "alternate_code": [
                  {
                    "server_version_id": 0,
                    "server_probe_code": "string",
                    "extension_version": 0
                  }
                ],
                "internal_name": "string"
```

(continues on next page)

(continued from previous page)

```
                }
            ]
        }
    ]
  }
 ]
}
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	Inline

Response Schema

Status Code **200**

Name	Type	Required	Restrictions	Description
» version	integer	false	none	Export version id
» items	*Query information*]	false	none	none
»» version	integer	false	none	Version of the alert template export
»» alert_templates	*Query information*]	false	none	none
»»» name	string	false	none	Name of the alert template. NOTE: This property is a mandatory parameter for a POST request.
»»» description	string	false	none	Description of the alert template. NOTE: This property is a mandatory parameter for a POST request.
»»» reference_id	string	false	none	Internal identifier for the alert template
»»» default_history_retention	integer	false	none	Defines the history retention period in days for this type of alerts. NOTE: This property is a mandatory parameter for a POST request.

continues on next page

Table 324 – continued from previous page

Name	Type	Required	Restrictions	Description
»»»» object_type	integer	false	none	Defines the type of the object to which this alert template applies. - A system-wide alert template is 50. - An alert template applicable to an agent is 100. - An alert template applicable to a managed server is 200. - An alert template applicable to a database for a particular monitored server is 300. - An alert template applicable to a schema on a particular database for the managed server is 400. - An alert template applicable to a table on a particular schema for the database on a monitored server is 500. - An alert template applicable to an index on a particular schema for the database on a monitored server is 600. - An alert template applicable to a sequence on a particular schema for the database on a monitored server is 700. - An alert template applicable to a function on a particular schema for the database on a monitored server is 800. NOTE: This property is a mandatory parameter for a POST request.

continues on next page

Table 324 – continued from previous page

Name	Type	Required	Restrictions	Description
»»»» sql	string	false	none	SQL query, possibly parameterized, to get a numeric value indicating the threshold reached. NOTE: This property is a mandatory parameter for a POST request.
»»»» applicable_on_server	string	false	none	Specifies the type of monitored server to which the template applies. Possible values are ALL, POSTGRES_SERVER, ADVANCED_SERVER. NOTE: This property is a mandatory parameter for a POST request.
»»»» default_check_frequency	integer	false	none	Specifies how frequently (in minutes) to check for the alert that this template defines. NOTE: This property is a mandatory parameter for a POST request.
»»»» threshold_unit	string	false	none	Units used to calculate the threshold, such as MB/kB/seconds/days/... This value is only used for the GUI.
»»»» is_system_template	boolean	false	none	Determines whether the given template is system-defined or user-defined. NOTE: This property is not applicable for POST/PUT requests.
»»»» is_auto_create	boolean	false	none	Determines whether an alert should be created as soon as an agent or server is added to PEM..
»»»» thresholds	[number]	false	none	Default threshold values. Values in the thresholds array must be in a sorted order based on the operator.
»»»» operator	string	false	none	Operator to compare threshold values
»»»» high_threshold_value	string	false	none	High threshold value
»»»» medium_threshold_value	string	false	none	Medium threshold value
»»»» low_threshold_value	string	false	none	Low threshold value

continues on next page

Table 324 – continued from previous page

Name	Type	Required	Restrictions	Description
»»»» info_sql	string	false	none	Defines the SQL query that provides detailed information about the alert on the dashboard.
»»»» probes	[allOf]	false	none	none

- allOf*

Name	Type	Required	Restrictions	Description
»»»»» *anonymous*	*CustomProbePostData*	false	none	none
»»»»»» probe_name	string	false	none	Name of the probe
»»»»»» collection_method	string	false	none	Defines the method of statistics collection. PEM supports three methods for custom made probes: SQL, WMI, or batch/shell script. 'w' stands for WMI script. This type of script only applies to the Windows platform, and the target_type is 100 (for agent). 'b' stands for batch/shell script. The 'target_type' for this script is always 100 (agent). This script behaves as a batch script on Windows and a shell script on other platforms. 's' stands for SQL script. The 'target_type' is always 200 (server), 300 (database), 400 (schema), 500 (table), 600 (index), 700 (sequence), 800 (function), 900 (view) or 1000 (extension).
»»»»»» enabled	boolean	false	none	Determines whether whether the probe is enabled/disabled by default.

continues on next page

Table 325 – continued from previous page

Name	Type	Required	Restrictions	Description
»»»»»»» target_type	integer	false	none	Defines the type of the object to which the probe applies. - A probe with target_type 100 is responsible for collecting statistics for an agent. - A probe with target_type 200 is responsible for collecting statistics for a monitored server. - A probe with target_type 300 is responsible for collecting statistics for a database. - A probe with target_type 400 is responsible for collecting statistics for a schema. - A probe with target_type 500 is responsible for collecting statistics for a table. - A probe with target_type 600 is responsible for collecting statistics for a index. - A probe with target_type 700 is responsible for collecting statistics for a sequence. - A probe with target_type 800 is responsible for collecting statistics for a function. - A probe with target_type 900 is responsible for collecting statistics for a view. - A probe with target_type 1000 is responsible for collecting statistics for a extension.
»»»»»»» any_server_version	boolean	false	none	Determines whether this probe applies to any version of a Postgres variant. For more details, please consult the documentation.

continues on next page

Table 325 – continued from previous page

Name	Type	Required	Restrictions	Description
»»»»»»» any_extension_version	boolean	false	none	Determines whether this probe applies to any version of a Extension variant. For more details, please consult the documentation.
»»»»»»» extension_name	string	false	none	Name of the extension. Valid names are edbspl , plpgsql , hstore , pldbgapi , edb_dblink_oci , edb_dblink_libpq , sslutils .
»»»»»»» interval	integer	false	none	Defines how frequently the probe collects statistics (in seconds).Must be greater than or equal to 10
»»»»»»» lifetime	integer	false	none	Defines the history data retention time (in days). Valid range is 1-365. This value does not have any effect if 'discard_history' is set to true.
»»»»»»» discard_history	boolean	false	none	Defines whether to discard history data. When 'discard_history' is set to false, no history data is stored for this probe.
»»»»»»» platform	string	false	none	Defines the platform where this probe can run. This value only applies to an agent-level probe.
»»»»»»» probe_code	string	false	none	A SQL/Batch script/Shell script/WMI script to collect the statistics as per definition of the code.
»»»»»»» probe_columns	[object]	false	none	none
»»»»»»»» pc_name	string	false	none	Column name

continues on next page

Table 325 – continued from previous page

Name	Type	Required	Restrictions	Description
»»»»»»»» pc_data_type	string	false	none	Data type of the column. Supported datatypes are: bigint char decimal double precision integer numeric real text tex tamp timestamptz
»»»»»»»» pc_unit	string	false	none	Returns the unit of the column.
»»»»»»»» pc_col_type	string	false	none	Returns the columns as part of the primary key of the internal table, or not. 'k' stands for a key column, and 'm' stands for a member/normal column.
»»»»»»»» pc_graphable	boolean	false	none	Determines whether the data of the column can be graphed on a chart.
»»»»»»»» pc_calc_pit	boolean	false	none	Determines whether the server calculates point-in-time for metric data for this column.
»»»»»»»» pc_pit_default	boolean	false	none	Determines whether metric data is stored by point-in-time by default. Please refer to the 'Custom Probes' section in the documentation for detailed information.
»»»»»»» alternate_code	*Query information*]	false	none	none
»»»»»»»» server_version_id	integer	false	none	Server version ID for the monitored server
»»»»»»»» server_probe_code	string	false	none	SQL query to run on the specified server version
»»»»»»»» extension_version	integer	false	none	Specifies the extension version. Valid for extension target type

- and*

Name	Type	Required	Restrictions	Description
»»»»» *anonymous*	object	false	none	none
»»»»»»» internal_name	string	false	none	Internal name of the probe from the system where probe was exported

Enumerated Values

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
pc_col_type	k

Note

This operation does not require authentication

Import Custom Probes

post*_probe_custom_import*

Code samples

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v16/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,
```

(continues on next page)

(continued from previous page)

```
  X-Auth-Token: string
}

r = requests.post(https://PEM-SERVER-IP/pem/api/v16/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,
      "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
};
```

(continues on next page)

(continued from previous page)

```
fetch(https://PEM-SERVER-IP/pem/api/v16/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/`

Imports custom probes using exported probes.

Body parameter

```
{
  "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
        }
```

(continues on next page)

(continued from previous page)

```
      ],
      "internal_name": "string"
    }
  ]
}
```

Parameters

Name	In	Type	Required	Description
X-Auth-Token	header	string	true	Token of authorization
body	body	*CustomProbeImportData*	true	none

Example responses

```
[
  {
    "name": "string",
    "status": "string",
    "msg": "string"
  }
]
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Status 200	Inline

Response Schema

Status Code **200**

Name	Type	Required	Restrictions	Description
» name	string	true	none	none
» status	string	true	none	none
» msg	string	true	none	none

Note

This operation does not require authentication

Probe Configuration

get_*probe_config_agent*{agent_id}

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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/v16/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/v16/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}`

Gets all the agent-level probes.

Parameters

Name	In	Type	Required	Description
agent_id	path	integer	true	Agent ID
X-Auth-Token	header	string	true	Token of authorization

Example responses

```
[
  {
    "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
  }
]
```


Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	Inline

Response Schema

Status Code **200**

Name	Type	Required	Restrictions	Description
anonymous	*Query information*]	false	none	none
» probe_id	integer	false	none	Unique ID of the probe
» probe_name	string	false	none	Name of the probe
» default_interval_min	integer	false	none	Defines the default interval for the probe to collect statistics (in minutes). Possible values are 0 to 2880.
» interval_min	integer	false	none	Defines how frequently the probe collects statistics (in minutes). Possible values are 0 to 2880.
» default_interval_sec	integer	false	none	Defines the default interval for the probe to collect statistics (in seconds). Possible values are 0 to 59.

continues on next page

Table 333 – continued from previous page

Name	Type	Required	Restrictions	Description
» interval_sec	integer	false	none	Defines how frequently the probe collects statistics (in seconds). Possible values are 0 to 59.
» default_interval	integer	false	none	Defines the default interval for the probe to collect statistics (in seconds).
» interval	integer	false	none	Defines how frequently the probe collects statistics (in seconds).Must be greater than or equal to 10
» default_lifetime	integer	false	none	Defines the default history data retention time (in days).
» lifetime	integer	false	none	Defines the history data retention time (in days). Valid range is 1-365. This value does not have any effect if 'discard_history' is set to true.
» default_enabled	boolean	false	none	Defines the default probe enabled/disabled value.
» enabled	boolean	false	none	Specifies whether whether the probe is enabled/disabled by default.

continues on next page

Table 333 – continued from previous page

Name	Type	Required	Restrictions	Description
» target_type_id_returned	integer	false	none	Defines the return target type value. - A probe with target_type 100 is responsible for collecting statistics for an agent. - A probe with target_type 200 is responsible for collecting statistics for a monitored server. - A probe with target_type 300 is responsible for collecting statistics for a database. - A probe with target_type 400 is responsible for collecting statistics for a schema. - A probe with target_type 500 is responsible for collecting statistics for a table. - A probe with target_type 600 is responsible for collecting statistics for a index. - A probe with target_type 700 is responsible for collecting statistics for a sequence. - A probe with target_type 800 is responsible for collecting statistics for a function. - A probe with target_type 900 is responsible for collecting statistics for a view.

Note

This operation does not require authentication

get_*probe_config_server*{server_id}

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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/v16/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/v16/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}`

Gets all the server-level probes.

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
X-Auth-Token	header	string	true	Token of authorization

Example responses

```
[
  {
    "probe_id": 0,
    "probe_name": "string",
    "default_interval_min": 0,
```

(continues on next page)

(continued from previous page)


```
    "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
  }
]
```


Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	Inline

Response Schema

Status Code **200**

Name	Type	Required	Restrictions	Description
anonymous	*Query information*]	false	none	none
» probe_id	integer	false	none	Unique ID of the probe
» probe_name	string	false	none	Name of the probe
» default_interval_min	integer	false	none	Defines the default interval for the probe to collect statistics (in minutes). Possible values are 0 to 2880.
» interval_min	integer	false	none	Defines how frequently the probe collects statistics (in minutes). Possible values are 0 to 2880.
» default_interval_sec	integer	false	none	Defines the default interval for the probe to collect statistics (in seconds). Possible values are 0 to 59.
» interval_sec	integer	false	none	Defines how frequently the probe collects statistics (in seconds). Possible values are 0 to 59.

continues on next page

Table 336 – continued from previous page

Name	Type	Required	Restrictions	Description
» default_interval	integer	false	none	Defines the default interval for the probe to collect statistics (in seconds).
» interval	integer	false	none	Defines how frequently the probe collects statistics (in seconds).Must be greater than or equal to 10
» default_lifetime	integer	false	none	Defines the default history data retention time (in days).
» lifetime	integer	false	none	Defines the history data retention time (in days). Valid range is 1-365. This value does not have any effect if 'discard_history' is set to true.
» default_enabled	boolean	false	none	Defines the default probe enabled/disabled value.
» enabled	boolean	false	none	Specifies whether whether the probe is enabled/disabled by default.

continues on next page

Table 336 – continued from previous page

Name	Type	Required	Restrictions	Description
» target_type_id_returned	integer	false	none	Defines the return target type value. - A probe with target_type 100 is responsible for collecting statistics for an agent. - A probe with target_type 200 is responsible for collecting statistics for a monitored server. - A probe with target_type 300 is responsible for collecting statistics for a database. - A probe with target_type 400 is responsible for collecting statistics for a schema. - A probe with target_type 500 is responsible for collecting statistics for a table. - A probe with target_type 600 is responsible for collecting statistics for a index. - A probe with target_type 700 is responsible for collecting statistics for a sequence. - A probe with target_type 800 is responsible for collecting statistics for a function. - A probe with target_type 900 is responsible for collecting statistics for a view.

Note

This operation does not require authentication

get_*probe_config_server*{server_id}_{probe_id}

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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/v16/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/v16/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}`

Gets server-level probe for specified probe ID.

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
probe_id	path	integer	true	Probe ID for which probe configuration will be returned
X-Auth-Token	header	string	true	Token of authorization

Example responses

```
{
  "probe_id": 0,
```

(continues on next page)

(continued from previous page)

```
  "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
}
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	*Probes*

Note

This operation does not require authentication

put_*probe_config_server*{server_id}_{probe_id}

Code samples

```
#  You can also use wget

curl -X PUT https://PEM-SERVER-IP/pem/api/v16/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/v16/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/v16/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}`

Update server-level probe configuration for specified probe ID.

Body parameter

```
{
  "interval_min": 0,
  "interval_sec": 0,
  "lifetime": 0,
  "enabled": true
}
```

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
probe_id	path	integer	true	Probe ID for which probe configuration will be returned
X-Auth-Token	header	string	true	Token of authorization
body	body	*ProbeConfigData*	true	none

Example responses

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Status 200	None

Response Schema

Note

This operation does not require authentication

get_*probe_config_server*{server_id}*database*{database_name}

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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/v16/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/v16/probe/config/server/{server_id}/database/
↪{database_name},
{
  method: GET,

  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
```

(continues on next page)

(continued from previous page)

```
    console.log(body);
});
```

`GET /probe/config/server/{server_id}/database/{database_name}`

Gets all the database-level probes.

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
database_name	path	string	true	Database name
X-Auth-Token	header	string	true	Token of authorization

Example responses


```
[
  {
    "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
  }
]
```


Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	Inline

Response Schema

Status Code **200**

Name	Type	Required	Restrictions	Description
anonymous	*Query information*]	false	none	none
» probe_id	integer	false	none	Unique ID of the probe
» probe_name	string	false	none	Name of the probe
» default_interval_min	integer	false	none	Defines the default interval for the probe to collect statistics (in minutes). Possible values are 0 to 2880.
» interval_min	integer	false	none	Defines how frequently the probe collects statistics (in minutes). Possible values are 0 to 2880.
» default_interval_sec	integer	false	none	Defines the default interval for the probe to collect statistics (in seconds). Possible values are 0 to 59.
» interval_sec	integer	false	none	Defines how frequently the probe collects statistics (in seconds). Possible values are 0 to 59.
» default_interval	integer	false	none	Defines the default interval for the probe to collect statistics (in seconds).
» interval	integer	false	none	Defines how frequently the probe collects statistics (in seconds).Must be greater than or equal to 10
» default_lifetime	integer	false	none	Defines the default history data retention time (in days).
» lifetime	integer	false	none	Defines the history data retention time (in days). Valid range is 1-365. This value does not have any effect if 'discard_history' is set to true.
» default_enabled	boolean	false	none	Defines the default probe enabled/disabled value.
» enabled	boolean	false	none	Specifies whether whether the probe is enabled/disabled by default.

continues on next page

Table 343 – continued from previous page

Name	Type	Required	Restrictions	Description
» target_type_id_returned	integer	false	none	Defines the return target type value. - A probe with target_type 100 is responsible for collecting statistics for an agent. - A probe with target_type 200 is responsible for collecting statistics for a monitored server. - A probe with target_type 300 is responsible for collecting statistics for a database. - A probe with target_type 400 is responsible for collecting statistics for a schema. - A probe with target_type 500 is responsible for collecting statistics for a table. - A probe with target_type 600 is responsible for collecting statistics for a index. - A probe with target_type 700 is responsible for collecting statistics for a sequence. - A probe with target_type 800 is responsible for collecting statistics for a function. - A probe with target_type 900 is responsible for collecting statistics for a view.

Note

This operation does not require authentication

get_*probe_config_server*{server_id}*database*{database_name}_{probe_id}

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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/v16/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/v16/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}`

Gets database-level probe for specified probe ID.

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
database_name	path	string	true	Database name
probe_id	path	integer	true	Probe ID for which probe configuration will be returned.
X-Auth-Token	header	string	true	Token of authorization

Example responses

```
{
  "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
}
```


Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	*Probes*

Note

This operation does not require authentication

put_*probe_config_server*{server_id}*database*{database_name}_{probe_id}

Code samples

```
#  You can also use wget

curl -X PUT https://PEM-SERVER-IP/pem/api/v16/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/v16/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/v16/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}`

Updates database-level probe configuration for specified probe ID.

Body parameter

```
{
  "interval_min": 0,
  "interval_sec": 0,
  "lifetime": 0,
  "enabled": true
}
```

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
database_name	path	string	true	Database name
probe_id	path	integer	true	Probe ID for which probe configuration will be returned
X-Auth-Token	header	string	true	Token of authorization
body	body	*ProbeConfigData*	true	none

Example responses

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Status 200	None

Response Schema

Note

This operation does not require authentication

get_*probe_config_server*{server_id}*database*{database_name}*schema*{schema_name}

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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/v16/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/v16/probe/config/server/{server_id}/database/
↪{database_name}/schema/{schema_name},
{
  method: GET,

  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
```

(continues on next page)

(continued from previous page)

```
    console.log(body);
});
```

`GET /probe/config/server/{server_id}/database/{database_name}/schema/{schema_name}`

Gets all the schema-level probes.

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
database_name	path	string	true	Database name
schema_name	path	string	true	Schema name
X-Auth-Token	header	string	true	Token of authorization

Example responses


```
[
  {
    "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
  }
]
```


Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	Inline

Response Schema

Status Code **200**

Name	Type	Required	Restrictions	Description
anonymous	*Query information*]	false	none	none
» probe_id	integer	false	none	Unique ID of the probe
» probe_name	string	false	none	Name of the probe
» default_interval_min	integer	false	none	Defines the default interval for the probe to collect statistics (in minutes). Possible values are 0 to 2880.
» interval_min	integer	false	none	Defines how frequently the probe collects statistics (in minutes). Possible values are 0 to 2880.
» default_interval_sec	integer	false	none	Defines the default interval for the probe to collect statistics (in seconds). Possible values are 0 to 59.
» interval_sec	integer	false	none	Defines how frequently the probe collects statistics (in seconds). Possible values are 0 to 59.
» default_interval	integer	false	none	Defines the default interval for the probe to collect statistics (in seconds).
» interval	integer	false	none	Defines how frequently the probe collects statistics (in seconds).Must be greater than or equal to 10
» default_lifetime	integer	false	none	Defines the default history data retention time (in days).
» lifetime	integer	false	none	Defines the history data retention time (in days). Valid range is 1-365. This value does not have any effect if 'discard_history' is set to true.
» default_enabled	boolean	false	none	Defines the default probe enabled/disabled value.
» enabled	boolean	false	none	Specifies whether whether the probe is enabled/disabled by default.

continues on next page

Table 350 – continued from previous page

Name	Type	Required	Restrictions	Description
» target_type_id_returned	integer	false	none	Defines the return target type value. - A probe with target_type 100 is responsible for collecting statistics for an agent. - A probe with target_type 200 is responsible for collecting statistics for a monitored server. - A probe with target_type 300 is responsible for collecting statistics for a database. - A probe with target_type 400 is responsible for collecting statistics for a schema. - A probe with target_type 500 is responsible for collecting statistics for a table. - A probe with target_type 600 is responsible for collecting statistics for a index. - A probe with target_type 700 is responsible for collecting statistics for a sequence. - A probe with target_type 800 is responsible for collecting statistics for a function. - A probe with target_type 900 is responsible for collecting statistics for a view.

Note

This operation does not require authentication

get_*probe_config_server*{server_id}*database*{database_name}*schema*{schema_name}_{probe_id}

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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/v16/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/v16/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}`

Gets schema-level probe for specified probe ID.

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
database_name	path	string	true	Database name
schema_name	path	string	true	Schema name
probe_id	path	integer	true	Probe ID for which probe configuration will be returned
X-Auth-Token	header	string	true	Token of authorization

Example responses


```
{
  "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
}
```


Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	*Probes*

Note

This operation does not require authentication

put_*probe_config_server*{server_id}*database*{database_name}*schema*{schema_name}_{probe_id}

Code samples

```
#  You can also use wget

curl -X PUT https://PEM-SERVER-IP/pem/api/v16/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/v16/probe/config/server/{server_id}/
```

(continues on next page)

(continued from previous page)

```
↪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/v16/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}`

Updates schema-level probe configuration for specified probe ID.

Body parameter

```
{
  "interval_min": 0,
  "interval_sec": 0,
  "lifetime": 0,
  "enabled": true
}
```

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
database_name	path	string	true	Database name
schema_name	path	string	true	Schema name
probe_id	path	integer	true	Probe ID for which probe configuration will be returned
X-Auth-Token	header	string	true	Token of authorization

continues on next page

Table 353 – continued from previous page

Name	In	Type	Required	Description
body	body	*ProbeConfig-Data*	true	none

Example responses

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Status 200	None

Response Schema

Note

This operation does not require authentication

get*_probe_config_server*{server_id}*database*{database_name}*schema*{schema_name}*table*{table_name}

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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/v16/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/v16/probe/config/server/{server_id}/database/
```

(continues on next page)

(continued from previous page)

```
↪{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}`

Gets all the table-level probes.

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
database_name	path	string	true	Database name
schema_name	path	string	true	Schema name
table_name	path	string	true	Table name
X-Auth-Token	header	string	true	Token of authorization

Example responses


```
[
  {
    "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
  }
]
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	Inline

Response Schema

Status Code **200**

Name	Type	Required	Restrictions	Description
anonymous	*Query information*]	false	none	none
» probe_id	integer	false	none	Unique ID of the probe
» probe_name	string	false	none	Name of the probe
» default_interval_min	integer	false	none	Defines the default interval for the probe to collect statistics (in minutes). Possible values are 0 to 2880.
» interval_min	integer	false	none	Defines how frequently the probe collects statistics (in minutes). Possible values are 0 to 2880.
» default_interval_sec	integer	false	none	Defines the default interval for the probe to collect statistics (in seconds). Possible values are 0 to 59.
» interval_sec	integer	false	none	Defines how frequently the probe collects statistics (in seconds). Possible values are 0 to 59.
» default_interval	integer	false	none	Defines the default interval for the probe to collect statistics (in seconds).
» interval	integer	false	none	Defines how frequently the probe collects statistics (in seconds).Must be greater than or equal to 10
» default_lifetime	integer	false	none	Defines the default history data retention time (in days).
» lifetime	integer	false	none	Defines the history data retention time (in days). Valid range is 1-365. This value does not have any effect if 'discard_history' is set to true.

continues on next page

Table 357 – continued from previous page

Name	Type	Required	Restrictions	Description
» default_enabled	boolean	false	none	Defines the default probe enabled/disabled value.
» enabled	boolean	false	none	Specifies whether whether the probe is enabled/disabled by default.
» target_type_id_returned	integer	false	none	Defines the return target type value. - A probe with target_type 100 is responsible for collecting statistics for an agent. - A probe with target_type 200 is responsible for collecting statistics for a monitored server. - A probe with target_type 300 is responsible for collecting statistics for a database. - A probe with target_type 400 is responsible for collecting statistics for a schema. - A probe with target_type 500 is responsible for collecting statistics for a table. - A probe with target_type 600 is responsible for collecting statistics for a index. - A probe with target_type 700 is responsible for collecting statistics for a sequence. - A probe with target_type 800 is responsible for collecting statistics for a function. - A probe with target_type 900 is responsible for collecting statistics for a view.

Note

This operation does not require authentication

get_*probe_config_server*{server_id}*database*{database_name}*schema*{schema_name}*table*{table_name}_{probe

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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/v16/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/v16/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}`

Gets table-level probe for specified probe ID.

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
database_name	path	string	true	Database name
schema_name	path	string	true	Schema name

continues on next page

Table 358 – continued from previous page

Name	In	Type	Required	Description
table_name	path	string	true	Table name
probe_id	path	integer	true	Probe ID for which probe configuration will be returned
X-Auth-Token	header	string	true	Token of authorization

Example responses

```
{
  "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
}
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	*Probes*

Note

This operation does not require authentication

get_*probe_config_server*{server_id}*database*{database_name}*schema*{schema_name}*index*{index_name}

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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/v16/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/v16/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}`

Gets all the index-level probes.

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
database_name	path	string	true	Database name
schema_name	path	string	true	Schema name
index_name	path	string	true	Index name
X-Auth-Token	header	string	true	Token of authorization

Example responses

```
[
  {
    "probe_id": 0,
    "probe_name": "string",
    "default_interval_min": 0,
    "interval_min": 0,
```

(continues on next page)

(continued from previous page)


```
    "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
  }
]
```


Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	Inline

Response Schema

Status Code **200**

Name	Type	Required	Restrictions	Description
anonymous	*Query information*]	false	none	none
» probe_id	integer	false	none	Unique ID of the probe
» probe_name	string	false	none	Name of the probe
» default_interval_min	integer	false	none	Defines the default interval for the probe to collect statistics (in minutes). Possible values are 0 to 2880.
» interval_min	integer	false	none	Defines how frequently the probe collects statistics (in minutes). Possible values are 0 to 2880.
» default_interval_sec	integer	false	none	Defines the default interval for the probe to collect statistics (in seconds). Possible values are 0 to 59.
» interval_sec	integer	false	none	Defines how frequently the probe collects statistics (in seconds). Possible values are 0 to 59.
» default_interval	integer	false	none	Defines the default interval for the probe to collect statistics (in seconds).

continues on next page

Table 362 – continued from previous page

Name	Type	Required	Restrictions	Description
» interval	integer	false	none	Defines how frequently the probe collects statistics (in seconds).Must be greater than or equal to 10
» default_lifetime	integer	false	none	Defines the default history data retention time (in days).
» lifetime	integer	false	none	Defines the history data retention time (in days). Valid range is 1-365. This value does not have any effect if 'discard_history' is set to true.
» default_enabled	boolean	false	none	Defines the default probe enabled/disabled value.
» enabled	boolean	false	none	Specifies whether whether the probe is enabled/disabled by default.

continues on next page

Table 362 – continued from previous page

Name	Type	Required	Restrictions	Description
» target_type_id_returned	integer	false	none	Defines the return target type value. - A probe with target_type 100 is responsible for collecting statistics for an agent. - A probe with target_type 200 is responsible for collecting statistics for a monitored server. - A probe with target_type 300 is responsible for collecting statistics for a database. - A probe with target_type 400 is responsible for collecting statistics for a schema. - A probe with target_type 500 is responsible for collecting statistics for a table. - A probe with target_type 600 is responsible for collecting statistics for a index. - A probe with target_type 700 is responsible for collecting statistics for a sequence. - A probe with target_type 800 is responsible for collecting statistics for a function. - A probe with target_type 900 is responsible for collecting statistics for a view.

Note

This operation does not require authentication

get_*probe_config_server*{server_id}*database*{database_name}*schema*{schema_name}*index*{index_name}_{prob

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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/v16/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/v16/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}`

Gets index level probe for specified probe ID.

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
database_name	path	string	true	Database name
schema_name	path	string	true	Schema name
index_name	path	string	true	Index name
probe_id	path	integer	true	Probe ID for which probe configuration will be returned

continues on next page

Table 363 – continued from previous page

Name	In	Type	Required	Description
X-Auth-Token	header	string	true	Token of authorization

Example responses


```
{
  "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
}
```


Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	*Probes*

Note

This operation does not require authentication

get_*probe_config_server*{server_id}*database*{database_name}*schema*{schema_name}*sequence*{sequence_nam

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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
```

(continues on next page)

(continued from previous page)

```
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v16/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/v16/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}`

Gets all the sequence-level probes.

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
database_name	path	string	true	Database name
schema_name	path	string	true	Schema name
sequence_name	path	string	true	Sequence name
X-Auth-Token	header	string	true	Token of authorization

Example responses

```
[
  {
    "probe_id": 0,
    "probe_name": "string",
    "default_interval_min": 0,
    "interval_min": 0,
    "default_interval_sec": 0,
    "interval_sec": 0,
```

(continues on next page)

(continued from previous page)

```
    "default_interval": 0,
    "interval": 0,
    "default_lifetime": 0,
    "lifetime": 0,
    "default_enabled": true,
    "enabled": true,
    "target_type_id_returned": 0
  }
]
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	Inline

Response Schema

Status Code **200**

Name	Type	Required	Restrictions	Description
anonymous	*Query information*]	false	none	none
» probe_id	integer	false	none	Unique ID of the probe
» probe_name	string	false	none	Name of the probe
» default_interval_min	integer	false	none	Defines the default interval for the probe to collect statistics (in minutes). Possible values are 0 to 2880.
» interval_min	integer	false	none	Defines how frequently the probe collects statistics (in minutes). Possible values are 0 to 2880.
» default_interval_sec	integer	false	none	Defines the default interval for the probe to collect statistics (in seconds). Possible values are 0 to 59.
» interval_sec	integer	false	none	Defines how frequently the probe collects statistics (in seconds). Possible values are 0 to 59.
» default_interval	integer	false	none	Defines the default interval for the probe to collect statistics (in seconds).

continues on next page

Table 367 – continued from previous page

Name	Type	Required	Restrictions	Description
» interval	integer	false	none	Defines how frequently the probe collects statistics (in seconds).Must be greater than or equal to 10
» default_lifetime	integer	false	none	Defines the default history data retention time (in days).
» lifetime	integer	false	none	Defines the history data retention time (in days). Valid range is 1-365. This value does not have any effect if 'discard_history' is set to true.
» default_enabled	boolean	false	none	Defines the default probe enabled/disabled value.
» enabled	boolean	false	none	Specifies whether whether the probe is enabled/disabled by default.

continues on next page

Table 367 – continued from previous page

Name	Type	Required	Restrictions	Description
» target_type_id_returned	integer	false	none	Defines the return target type value. - A probe with target_type 100 is responsible for collecting statistics for an agent. - A probe with target_type 200 is responsible for collecting statistics for a monitored server. - A probe with target_type 300 is responsible for collecting statistics for a database. - A probe with target_type 400 is responsible for collecting statistics for a schema. - A probe with target_type 500 is responsible for collecting statistics for a table. - A probe with target_type 600 is responsible for collecting statistics for a index. - A probe with target_type 700 is responsible for collecting statistics for a sequence. - A probe with target_type 800 is responsible for collecting statistics for a function. - A probe with target_type 900 is responsible for collecting statistics for a view.

Note

This operation does not require authentication

get_*probe_config_server*{server_id}*database*{database_name}*schema*{schema_name}*sequence*{sequence_nam

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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/v16/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/v16/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) {
    console.log(body);
});
```

`GET /probe/config/server/{server_id}/database/{database_name}/schema/{schema_name}/sequence/{sequence_name}/{probe_id}`

Gets sequence-level probe for specified probe ID.

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
database_name	path	string	true	Database name
schema_name	path	string	true	Schema name
sequence_name	path	string	true	Sequence name
probe_id	path	integer	true	Probe ID for which probe configuration will be returned

continues on next page

Table 368 – continued from previous page

Name	In	Type	Required	Description
X-Auth-Token	header	string	true	Token of authorization

Example responses

```
{
  "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
}
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	*Probes*

Note

This operation does not require authentication

get_*probe_config_server*{server_id}*database*{database_name}*schema*{schema_name}*function*{function_name}_

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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
```

(continues on next page)

(continued from previous page)

```
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v16/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/v16/probe/config/server/{server_id}/database/
↪{database_name}/schema/{schema_name}/function/{function_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}/function/{function_name}/`

Gets all the function-level probes.

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
database_name	path	string	true	Database name
schema_name	path	string	true	Schema name
function_name	path	string	true	Function name
X-Auth-Token	header	string	true	Token of authorization

Example responses

```
[
  {
    "probe_id": 0,
    "probe_name": "string",
    "default_interval_min": 0,
    "interval_min": 0,
    "default_interval_sec": 0,
    "interval_sec": 0,
```

(continues on next page)

(continued from previous page)

```
    "default_interval": 0,
    "interval": 0,
    "default_lifetime": 0,
    "lifetime": 0,
    "default_enabled": true,
    "enabled": true,
    "target_type_id_returned": 0
  }
]
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	Inline

Response Schema

Status Code **200**

Name	Type	Required	Restrictions	Description
anonymous	*Query information*]	false	none	none
» probe_id	integer	false	none	Unique ID of the probe
» probe_name	string	false	none	Name of the probe
» default_interval_min	integer	false	none	Defines the default interval for the probe to collect statistics (in minutes). Possible values are 0 to 2880.
» interval_min	integer	false	none	Defines how frequently the probe collects statistics (in minutes). Possible values are 0 to 2880.
» default_interval_sec	integer	false	none	Defines the default interval for the probe to collect statistics (in seconds). Possible values are 0 to 59.
» interval_sec	integer	false	none	Defines how frequently the probe collects statistics (in seconds). Possible values are 0 to 59.
» default_interval	integer	false	none	Defines the default interval for the probe to collect statistics (in seconds).

continues on next page

Table 372 – continued from previous page

Name	Type	Required	Restrictions	Description
» interval	integer	false	none	Defines how frequently the probe collects statistics (in seconds).Must be greater than or equal to 10
» default_lifetime	integer	false	none	Defines the default history data retention time (in days).
» lifetime	integer	false	none	Defines the history data retention time (in days). Valid range is 1-365. This value does not have any effect if 'discard_history' is set to true.
» default_enabled	boolean	false	none	Defines the default probe enabled/disabled value.
» enabled	boolean	false	none	Specifies whether whether the probe is enabled/disabled by default.

continues on next page

Table 372 – continued from previous page

Name	Type	Required	Restrictions	Description
» target_type_id_returned	integer	false	none	Defines the return target type value. - A probe with target_type 100 is responsible for collecting statistics for an agent. - A probe with target_type 200 is responsible for collecting statistics for a monitored server. - A probe with target_type 300 is responsible for collecting statistics for a database. - A probe with target_type 400 is responsible for collecting statistics for a schema. - A probe with target_type 500 is responsible for collecting statistics for a table. - A probe with target_type 600 is responsible for collecting statistics for a index. - A probe with target_type 700 is responsible for collecting statistics for a sequence. - A probe with target_type 800 is responsible for collecting statistics for a function. - A probe with target_type 900 is responsible for collecting statistics for a view.

Note

This operation does not require authentication

get_*probe_config_server*{server_id}*database*{database_name}*schema*{schema_name}*function*{function_name}_

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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/v16/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/v16/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}`

Gets function-level probe for specified probe ID.

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
database_name	path	string	true	Database name
schema_name	path	string	true	Schema name
function_name	path	string	true	Function name
probe_id	path	integer	true	Probe ID for which probe configuration will be returned

continues on next page

Table 373 – continued from previous page

Name	In	Type	Required	Description
X-Auth-Token	header	string	true	Token of authorization

Example responses

```
{
  "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
}
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	*Probes*

Note

This operation does not require authentication

get_*probe_config_server*{server_id}*database*{database_name}*schema*{schema_name}*view*{view_name}

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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
```

(continues on next page)

(continued from previous page)

```
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v16/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/v16/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}`

Gets all the view-level probes.

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
database_name	path	string	true	Database name
schema_name	path	string	true	Schema name
view_name	path	string	true	View name
X-Auth-Token	header	string	true	Token of authorization

Example responses

```
[
  {
    "probe_id": 0,
    "probe_name": "string",
    "default_interval_min": 0,
    "interval_min": 0,
    "default_interval_sec": 0,
    "interval_sec": 0,
    "default_interval": 0,
```

(continues on next page)

(continued from previous page)

```
    "interval": 0,
    "default_lifetime": 0,
    "lifetime": 0,
    "default_enabled": true,
    "enabled": true,
    "target_type_id_returned": 0
  }
]
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	Inline

Response Schema

Status Code **200**

Name	Type	Required	Restrictions	Description
anonymous	*Query information*]	false	none	none
» probe_id	integer	false	none	Unique ID of the probe
» probe_name	string	false	none	Name of the probe
» default_interval_min	integer	false	none	Defines the default interval for the probe to collect statistics (in minutes). Possible values are 0 to 2880.
» interval_min	integer	false	none	Defines how frequently the probe collects statistics (in minutes). Possible values are 0 to 2880.
» default_interval_sec	integer	false	none	Defines the default interval for the probe to collect statistics (in seconds). Possible values are 0 to 59.
» interval_sec	integer	false	none	Defines how frequently the probe collects statistics (in seconds). Possible values are 0 to 59.
» default_interval	integer	false	none	Defines the default interval for the probe to collect statistics (in seconds).

continues on next page

Table 377 – continued from previous page

Name	Type	Required	Restrictions	Description
» interval	integer	false	none	Defines how frequently the probe collects statistics (in seconds).Must be greater than or equal to 10
» default_lifetime	integer	false	none	Defines the default history data retention time (in days).
» lifetime	integer	false	none	Defines the history data retention time (in days). Valid range is 1-365. This value does not have any effect if 'discard_history' is set to true.
» default_enabled	boolean	false	none	Defines the default probe enabled/disabled value.
» enabled	boolean	false	none	Specifies whether whether the probe is enabled/disabled by default.

continues on next page

Table 377 – continued from previous page

Name	Type	Required	Restrictions	Description
» target_type_id_returned	integer	false	none	Defines the return target type value. - A probe with target_type 100 is responsible for collecting statistics for an agent. - A probe with target_type 200 is responsible for collecting statistics for a monitored server. - A probe with target_type 300 is responsible for collecting statistics for a database. - A probe with target_type 400 is responsible for collecting statistics for a schema. - A probe with target_type 500 is responsible for collecting statistics for a table. - A probe with target_type 600 is responsible for collecting statistics for a index. - A probe with target_type 700 is responsible for collecting statistics for a sequence. - A probe with target_type 800 is responsible for collecting statistics for a function. - A probe with target_type 900 is responsible for collecting statistics for a view.

Note

This operation does not require authentication

get_*probe_config_server*{server_id}*database*{database_name}*schema*{schema_name}*view*{view_name}_{probe_

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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/v16/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/v16/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}`

Gets view-level probe for specified probe ID.

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
database_name	path	string	true	Database name
schema_name	path	string	true	Schema name
view_name	path	string	true	View name
probe_id	path	integer	true	Probe ID for which probe configuration will be returned

continues on next page

Table 378 – continued from previous page

Name	In	Type	Required	Description
X-Auth-Token	header	string	true	Token of authorization

Example responses

```
{
  "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
}
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	*Probes*

Note

This operation does not require authentication

Copy Probe

post_*probe_copy_agent*{agent_id}

Code samples

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v16/probe/copy/agent/{agent_id} \
  -H Content-Type: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Content-Type: application/json,
```

(continues on next page)

(continued from previous page)

```
  X-Auth-Token: string
}

r = requests.post(https://PEM-SERVER-IP/pem/api/v16/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/v16/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}`

Copies agent-level probe to another agent.

Body parameter

```
[
  {
    "type": "string",
    "agent_id": 0
  }
]
```

Parameters

Name	In	Type	Required	Description
agent_id	path	integer	true	Agent ID
X-Auth-Token	header	string	true	Token of authorization

continues on next page

Table 380 – continued from previous page

Name	In	Type	Required	Description
body	body	*CopyAgent-Data*	true	none

Detailed descriptions

body: NOTE:

- **type** value must be 'agent'

Example responses

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Status 200	None

Response Schema

> **Note**
>
> This operation does not require authentication

post_*probe_copy_server*{server_id}

Code samples

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v16/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/v16/probe/copy/server/{server_id},␣
↪headers = headers)

print(r.json())
```

```
const inputBody = [
  {
```

(continues on next page)

(continued from previous page)

```
    "type": "string",
    "server_id": 0
  }
];
const headers = {
  Content-Type:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v16/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}`

Copies server-level probe to another server.

Body parameter

```
[
  {
    "type": "string",
    "server_id": 0
  }
]
```

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
X-Auth-Token	header	string	true	Token of authorization
body	body	*CopyServer-Data*	true	none

Detailed descriptions

body: NOTE:

- **type** value must be 'server'

Example responses

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Status 200	None

Response Schema

> **Note**
>
> This operation does not require authentication

post_*probe_copy_server*{server_id}*database*{database_name}

Code samples

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v16/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/v16/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/v16/probe/copy/server/{server_id}/database/
↪{database_name},
{
```

(continues on next page)

(continued from previous page)

```
  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}`

Copies database-level probe to another database.

Body parameter

```
[
  {
    "type": "string",
    "server_id": 0,
    "database_name": "string"
  }
]
```

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
database_name	path	string	true	Database name
X-Auth-Token	header	string	true	Token of authorization
body	body	*Copy-Database-Data*	true	none

Detailed descriptions

body: NOTE:

- **type** value must be 'database'

Example responses

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Status 200	None

Response Schema

> **Note**
>
> This operation does not require authentication

post_*probe_copy_server*{server_id}*database*{database_name}*schema*{schema_name}

Code samples

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v16/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/v16/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/v16/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);
```

(continues on next page)

(continued from previous page)

```
});
```

`POST /probe/copy/server/{server_id}/database/{database_name}/schema/{schema_name}`

Copies schema-level probe to another schema.

Body parameter

```
[
  {
    "type": "string",
    "server_id": 0,
    "database_name": "string",
    "schema_name": "string"
  }
]
```

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
database_name	path	string	true	Database name
schema_name	path	string	true	Schema name
X-Auth-Token	header	string	true	Token of authorization
body	body	*CopySchemaData*	true	none

Detailed descriptions

body: NOTE:

- **type** value must be 'schema'

Example responses

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Status 200	None

Response Schema

Note

This operation does not require authentication

Probe data

get_*probe_data_agent*{agent_id}_{probe_id}

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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/v16/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/v16/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}`

Gets `Probe data` for agent.

Parameters

Name	In	Type	Required	Description
agent_id	path	integer	true	Agent ID
probe_id	path	integer	true	Probe ID for which data will be returned
X-Auth-Token	header	string	true	Token of authorization

Example responses

```
[
  {}
]
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	Inline

Response Schema

Note

This operation does not require authentication

get_*probe_data_server*{server_id}_{probe_id}

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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/v16/probe/data/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/v16/probe/data/server/{server_id}/{probe_id},
{
  method: GET,
```

(continues on next page)

(continued from previous page)

```
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

`GET /probe/data/server/{server_id}/{probe_id}`

Gets `Probe data` for server.

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
probe_id	path	integer	true	Probe ID for which data will be returned
X-Auth-Token	header	string	true	Token of authorization

Example responses

```
[
  {}
]
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	Inline

Response Schema

Note

This operation does not require authentication

get_*probe_data_server*{server_id}*database*{database_name}_{probe_id}

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/probe/data/server/{server_id}/database/
↪{database_name}/{probe_id} \
```

(continues on next page)

(continued from previous page)

```
  -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/v16/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/v16/probe/data/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/data/server/{server_id}/database/{database_name}/{probe_id}`

Gets `Probe data` for database.

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
database_name	path	string	true	Database name
probe_id	path	integer	true	Probe ID for which data will be returned
X-Auth-Token	header	string	true	Token of authorization

Example responses

```
[
  {}
]
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	Inline

Response Schema

Note

This operation does not require authentication

get_*probe_data_server*{server_id}*database*{database_name}*schema*{schema_name}_{probe_id}

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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/v16/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/v16/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) {
```

(continues on next page)

(continued from previous page)

```
    console.log(body);
});
```

`GET /probe/data/server/{server_id}/database/{database_name}/schema/{schema_name}/{probe_id}`

Gets `Probe data` for schema.

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
database_name	path	string	true	Database name
schema_name	path	string	true	Schema name
probe_id	path	integer	true	Probe ID for which data will be returned
X-Auth-Token	header	string	true	Token of authorization

Example responses

```
[
  {}
]
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	Inline

Response Schema

Note

This operation does not require authentication

get_*probe_data_server*{server_id}*database*{database_name}*schema*{schema_name}*table*{table_name}_{probe_i

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v16/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/v16/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}`

Gets `Probe data` for table.

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
database_name	path	string	true	Database name
schema_name	path	string	true	Schema name
table_name	path	string	true	Table name
probe_id	path	integer	true	Probe ID for which data will be returned
X-Auth-Token	header	string	true	Token of authorization

Example responses

```
[
  {}
]
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	Inline

Response Schema

Note

This operation does not require authentication

get_*probe_data_server*{server_id}*database*{database_name}*schema*{schema_name}*index*{index_name}_{probe_

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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/v16/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
};

fetch(https://PEM-SERVER-IP/pem/api/v16/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();
```

(continues on next page)

(continued from previous page)

```
}).then(function(body) {
    console.log(body);
});
```

```
GET /probe/data/server/{server_id}/database/{database_name}/schema/{schema_name}/
index/{index_name}/{probe_id}
```

Gets `Probe data` for index.

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
database_name	path	string	true	Database name
schema_name	path	string	true	Schema name
index_name	path	string	true	Index name
probe_id	path	integer	true	Probe ID for which data will be returned
X-Auth-Token	header	string	true	Token of authorization

Example responses

```
[
  {}
]
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	Inline

Response Schema

Note

This operation does not require authentication

get_*probe_data_server*{server_id}*database*{database_name}*schema*{schema_name}*sequence*{sequence_name}

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/probe/data/server/{server_id}/database/
↪{database_name}/schema/{schema_name}/sequence/{sequence_name}/{probe_id} \
```

(continues on next page)

(continued from previous page)

```
  -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/v16/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/v16/probe/data/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) {
    console.log(body);
});
```

`GET /probe/data/server/{server_id}/database/{database_name}/schema/{schema_name}/sequence/{sequence_name}/{probe_id}`

Gets `Probe data` for sequence.

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
database_name	path	string	true	Database name
schema_name	path	string	true	Schema name
sequence_name	path	string	true	Sequence name
probe_id	path	integer	true	Probe ID for which data will be returned
X-Auth-Token	header	string	true	Token of authorization

Example responses

```
[
  {}
]
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	Inline

Response Schema

Note

This operation does not require authentication

get_*probe_data_server*{server_id}*database*{database_name}*schema*{schema_name}*view*{view_name}_{probe_id

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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/v16/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/v16/probe/data/server/{server_id}/database/
↪{database_name}/schema/{schema_name}/view/{view_name}/{probe_id},
{
```

(continues on next page)

(continued from previous page)

```
  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}/view/{view_name}/{probe_id}`

Gets `Probe data` for view.

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
database_name	path	string	true	Database name
schema_name	path	string	true	Schema name
view_name	path	string	true	View name
probe_id	path	integer	true	Probe ID for which data will be returned
X-Auth-Token	header	string	true	Token of authorization

Example responses

```
[
  {}
]
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	Inline

Response Schema

Note

This operation does not require authentication

get_*probe_data_server*{server_id}*database*{database_name}*schema*{schema_name}*function*{function_name}_{p

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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/v16/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/v16/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}`

Gets `Probe data` for function.

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
database_name	path	string	true	Database name
schema_name	path	string	true	Schema name
function_name	path	string	true	Function name
probe_id	path	integer	true	Probe ID for which data will be returned

continues on next page

Table 404 – continued from previous page

Name	In	Type	Required	Description
X-Auth-Token	header	string	true	Token of authorization

Example responses

```
[
  {}
]
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	Inline

Response Schema

Note

This operation does not require authentication

Probe history

get_*probe_history_agent*{agent_id}*from*{start_time}*to*{end_time}_{probe_id}

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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/v16/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/v16/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}`

Gets `Probe history` for agent.

Parameters

Name	In	Type	Required	Description
agent_id	path	integer	true	Agent ID
start_time	path	string	true	Start time in ISO-8601 format to get probe history
end_time	path	string	true	End time in ISO-8601 format to get probe history
probe_id	path	integer	true	Probe ID for which data will be returned
X-Auth-Token	header	string	true	Token of authorization

Example responses

```
[
  {}
]
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	Inline

Response Schema

> **Note**
>
> This operation does not require authentication

get_*probe_history_server*{server_id}*from*{start_time}*to*{end_time}_{probe_id}

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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/v16/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/v16/probe/history/server/{server_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/server/{server_id}/from/{start_time}/to/{end_time}/{probe_id}`

Gets `Probe history` for server.

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
start_time	path	string	true	Start time in ISO-8601 format to get probe history
end_time	path	string	true	End time in ISO-8601 format to get probe history
probe_id	path	integer	true	Probe ID for which data will be returned
X-Auth-Token	header	string	true	Token of authorization

Example responses

```
[
  {}
]
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	Inline

Response Schema

Note

This operation does not require authentication

get_*probe_history_server*{server_id}*database*{database_name}*from*{start_time}*to*{end_time}_{probe_id}

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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
}
```

(continues on next page)

(continued from previous page)

```
r = requests.get(https://PEM-SERVER-IP/pem/api/v16/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/v16/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}`

Gets `Probe history` for database.

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
database_name	path	string	true	Database name
start_time	path	string	true	Start time in ISO-8601 format to get probe history
end_time	path	string	true	End time in ISO-8601 format to get probe history
probe_id	path	integer	true	Probe ID for which data will be returned
X-Auth-Token	header	string	true	Token of authorization

Example responses

```
[
  {}
]
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	Inline

Response Schema

Note

This operation does not require authentication

get_*probe_history_server*{server_id}*database*{database_name}*schema*{schema_name}*from*{start_time}*to*{end_ti

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/probe/history/server/{server_id}/
↪database/{database_name}/schema/{schema_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/v16/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/v16/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) {
```

(continues on next page)

(continued from previous page)

```
    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}
```

Gets `Probe history` for schema.

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
database_name	path	string	true	Database name
schema_name	path	string	true	Schema name
start_time	path	string	true	Start time in ISO-8601 format to get probe history
end_time	path	string	true	End time in ISO-8601 format to get probe history
probe_id	path	integer	true	Probe ID for which data will be returned
X-Auth-Token	header	string	true	Token of authorization

Example responses

```
[
  {}
]
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	Inline

Response Schema

Note

This operation does not require authentication

get_*probe_history_server*{server_id}*database*{database_name}*schema*{schema_name}*table*{table_name}*from*{st

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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/v16/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/v16/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}`

Gets `Probe history` for table.

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
database_name	path	string	true	Database name
schema_name	path	string	true	Schema name
table_name	path	string	true	Table name

continues on next page

Table 414 – continued from previous page

Name	In	Type	Required	Description
start_time	path	string	true	Start time in ISO-8601 format to get probe history
end_time	path	string	true	End time in ISO-8601 format to get probe history
probe_id	path	integer	true	Probe ID for which data will be returned
X-Auth-Token	header	string	true	Token of authorization

Example responses

```
[
  {}
]
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	Inline

Response Schema

Note

This operation does not require authentication

get_*probe_history_server*{server_id}*database*{database_name}*schema*{schema_name}*index*{index_name}*from*{

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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 = {
  Accept: application/json,
  X-Auth-Token: string
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v16/probe/history/server/{server_id}/
```

(continues on next page)

(continued from previous page)

```
↪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/v16/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}`

Gets `Probe history` for index.

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
database_name	path	string	true	Database name
schema_name	path	string	true	Schema name
index_name	path	string	true	Index name
start_time	path	string	true	Start time in ISO-8601 format to get probe history
end_time	path	string	true	End time in ISO-8601 format to get probe history
probe_id	path	integer	true	Probe ID for which data will be returned
X-Auth-Token	header	string	true	Token of authorization

Example responses

```
[
  {}
]
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	Inline

Response Schema

Note

This operation does not require authentication

get_*probe_history_server*{server_id}*database*{database_name}*schema*{schema_name}*sequence*{sequence_nam

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v16/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/v16/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
})
```

(continues on next page)

(continued from previous page)

```
.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}`

Gets `Probe history` for sequence.

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
database_name	path	string	true	Database name
schema_name	path	string	true	Schema name
sequence_name	path	string	true	Sequence name
start_time	path	string	true	Start time in ISO-8601 format to get probe history
end_time	path	string	true	End time in ISO-8601 format to get probe history
probe_id	path	integer	true	Probe ID for which data will be returned
X-Auth-Token	header	string	true	Token of authorization

Example responses

```
[
  {}
]
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	Inline

Response Schema

Note

This operation does not require authentication

get_*probe_history_server*{server_id}*database*{database_name}*schema*{schema_name}*view*{view_name}*from*{st

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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
}

r = requests.get(https://PEM-SERVER-IP/pem/api/v16/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/v16/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}`

Gets `Probe history` for view.

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
database_name	path	string	true	Database name
schema_name	path	string	true	Schema name
view_name	path	string	true	View name

continues on next page

Table 420 – continued from previous page

Name	In	Type	Required	Description
start_time	path	string	true	Start time in ISO-8601 format to get probe history
end_time	path	string	true	End time in ISO-8601 format to get probe history
probe_id	path	integer	true	Probe ID for which data will be returned
X-Auth-Token	header	string	true	Token of authorization

Example responses

```
[
  {}
]
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	Inline

Response Schema

Note

This operation does not require authentication

get_*probe_history_server*{server_id}*database*{database_name}*schema*{schema_name}*function*{function_name}*f*

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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/v16/probe/history/server/{server_id}/
```

(continues on next page)

(continued from previous page)

```
↪database/{database_name}/schema/{schema_name}/function/{function_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/v16/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}`

Gets `Probe history` for function.

Parameters

Name	In	Type	Required	Description
server_id	path	integer	true	Server ID
database_name	path	string	true	Database name
schema_name	path	string	true	Schema name
function_name	path	string	true	Function name
start_time	path	string	true	Start time in ISO-8601 format to get probe history
end_time	path	string	true	End time in ISO-8601 format to get probe history
probe_id	path	integer	true	Probe ID for which data will be returned
X-Auth-Token	header	string	true	Token of authorization

Example responses

```
[
  {}
]
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Successful response	Inline

Response Schema

> **Note**
>
> This operation does not require authentication

Webhook

get_*webhook*

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/webhook/ \
  -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/v16/webhook/, headers = headers)

print(r.json())
```

```
const headers = {
  Accept:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v16/webhook/,
{
  method: GET,

  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
```

(continues on next page)

(continued from previous page)

```
});
```

```
GET /webhook/
```

- Retrieve the webhook list.*

Parameters

Name	In	Type	Required	Description
X-Auth-Token	header	string	true	Token of authorization

Example responses

```
[
  {
    "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"
  }
]
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Status 200	Inline

Response Schema

Status Code **200**

Name	Type	Required	Restrictions	Description
anonymous	*Query information*]	false	none	none
» id	integer	false	none	Webhook ID
» name	string	false	none	Webhook name
» url	string	false	none	Webhook URL
» enabled	boolean	false	none	Determines whether the webhook is enabled.
» method	string	false	none	Request method
» http_headers	*Query information*]	false	none	List of the HTTP header key/value pairs to be used by the webhook
»» http_header_id	integer	false	none	ID of the HTTP header
»» http_header_key	string	false	none	Name of the HTTP header key
»» http_header_value	string	false	none	Name of the HTTP header key
» payload_template	string	false	none	Payload template. Below is payload example for slack.
» low_alert	boolean	false	none	Determines whether the low alert is enabled.
» med_alert	boolean	false	none	Determines whether the medium alert is enabled.
» high_alert	boolean	false	none	Determines whether the high alert is enabled.
» cleared_alert	boolean	false	none	Determines whether the cleared alert is enabled.
» payload_type	string	false	none	Payload type

Enumerated Values

Property	Value
method	POST
method	PUT
payload_type	ALERT
payload_type	JOB

Note

This operation does not require authentication

post_*webhook*

Code samples

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v16/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/v16/webhook/, headers = headers)

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/v16/webhook/,
{
  method: POST,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

`POST /webhook/`

- Add a new webhook.*

Body parameter

```
{
  "name": "string",
  "url": "string",
```

(continues on next page)

(continued from previous page)

```
  "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"
}
```

Parameters

Name	In	Type	Required	Description
X-Auth-Token	header	string	true	Token of authorization
body	body	*Webhook-PostData*	true	none

Example responses

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Webhook created successfully.	None

Response Schema

> **Note**
>
> This operation does not require authentication

get_*webhook*{webhook_id}

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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/v16/webhook/{webhook_id}, headers =␣
↪headers)

print(r.json())
```

```
const headers = {
  Accept:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v16/webhook/{webhook_id},
{
  method: GET,

  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

`GET /webhook/{webhook_id}`

- Retrieve information for the webhook, identified by the webhook_id.*

Parameters

Name	In	Type	Required	Description
webhook_id	path	string	true	none
X-Auth-Token	header	string	true	Token of authorization

Example responses

```
{
  "id": 0,
  "name": "string",
  "url": "string",
  "enabled": true,
  "method": "POST",
  "http_headers": [
    {
      "http_header_id": 0,
      "http_header_key": "string",
      "http_header_value": "string"
```

(continues on next page)

(continued from previous page)

```
    }
  ],
  "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"
}
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Status 200	*Webhooks*

Note

This operation does not require authentication

put_*webhook*{webhook_id}

Code samples

```
#  You can also use wget

curl -X PUT https://PEM-SERVER-IP/pem/api/v16/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/v16/webhook/{webhook_id}, headers =␣
↪headers)

print(r.json())
```

```
const inputBody = {
  "name": "string",
  "url": "string",
  "enabled": true,
  "method": "POST",
```

(continues on next page)

(continued from previous page)

```
  "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
};
const headers = {
  Content-Type:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v16/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}`

- Update the webhook information.*

Body parameter

```
{
  "name": "string",
```

(continues on next page)

(continued from previous page)

```
  "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
}
```

Parameters

Name	In	Type	Required	Description
webhook_id	path	string	true	none
X-Auth-Token	header	string	true	Token of authorization
body	body	*WebhookPut-Data*	true	none

Example responses

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Webhook updated successfully.	None

Response Schema

> **Note**
>
> This operation does not require authentication

delete_*webhook*{webhook_id}

Code samples

```
#  You can also use wget

curl -X DELETE https://PEM-SERVER-IP/pem/api/v16/webhook/{webhook_id} \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  X-Auth-Token: string
}

r = requests.delete(https://PEM-SERVER-IP/pem/api/v16/webhook/{webhook_id}, headers =␣
↪headers)

print(r.json())
```

```
const headers = {
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v16/webhook/{webhook_id},
{
  method: DELETE,

  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

`DELETE /webhook/{webhook_id}`

- Delete the webhook.*

Parameters

Name	In	Type	Required	Description
webhook_id	path	string	true	none
X-Auth-Token	header	string	true	Token of authorization

Example responses

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Webhook deleted successfully	None

Response Schema

> **Note**
>
> This operation does not require authentication

Email Groups

get__email_groups

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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/v16/email/groups, headers = headers)

print(r.json())
```

```
const headers = {
  Accept:application/json,
  X-Auth-Token:string
};
```

(continues on next page)

(continued from previous page)

```
fetch(https://PEM-SERVER-IP/pem/api/v16/email/groups,
{
  method: GET,

  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

`GET /email/groups`

- Retrieve the email groups list.*

Parameters

Name	In	Type	Required	Description
X-Auth-Token	header	string	true	Token of authorization

Example responses

```
[
  {
    "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"
  }
]
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Status 200	Inline

Response Schema

Status Code **200**

Name	Type	Required	Restrictions	Description
anonymous	*Query information*]	false	none	none
» id	integer	false	none	Email group ID
» oid	integer	false	none	Email group options ID
» gid	integer	false	none	Email group ID
» name	string	false	none	Email group name
» to_addr	string	false	none	Email group options receiver address
» cc_addr	string	false	none	Email group options CC address
» bcc_addr	string	false	none	Email group options BCC address
» from_addr	string	false	none	Email group options sender address
» reply_to_addr	string	false	none	Email group options reply to address
» subject_prefix	string	false	none	Email group options subject prefix
» from_time	string	false	none	Email group options from time to select time range for members
» to_time	string	false	none	Email group options to time to select time range for members

Note

This operation does not require authentication

post__email_groups

Code samples

```
#  You can also use wget

curl -X POST https://PEM-SERVER-IP/pem/api/v16/email/groups \
  -H Content-Type: application/json \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  Content-Type: application/json,
  X-Auth-Token: string
```

(continues on next page)

(continued from previous page)

```
}

r = requests.post(https://PEM-SERVER-IP/pem/api/v16/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/v16/email/groups,
{
  method: POST,
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

`POST /email/groups`

- Add a new email group.*

Body parameter

```
{
  "name": "string",
  "options": [
    {
      "to_addr": "string",
      "cc_addr": "string",
      "bcc_addr": "string",
      "from_addr": "string",
      "reply_to_addr": "string",
```

(continues on next page)

(continued from previous page)

```
      "subject_prefix": "string",
      "from_time": "00:00:00",
      "to_time": "23:59:59"
    }
  ]
}
```

Parameters

Name	In	Type	Required	Description
X-Auth-Token	header	string	true	Token of authorization
body	body	*EmailGroupsPostData*	true	none

Example responses

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Email group created successfully	None

Response Schema

> **Note**
>
> This operation does not require authentication

get_*email_groups*{email_group_id}

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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/v16/email/groups/{email_group_id},␣
```

(continues on next page)

(continued from previous page)

```
↪headers = headers)

print(r.json())
```

```
const headers = {
  Accept:application/json,
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v16/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}`

- Retrieve information for the email group, identified by the email_group_id.*

Parameters

Name	In	Type	Required	Description
email_group_id	path	string	true	none
X-Auth-Token	header	string	true	Token of authorization

Example responses

```
{
  "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"
}
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Status 200	*EmailGroups*

Note

This operation does not require authentication

put_*email_groups*{email_group_id}

Code samples

```
#  You can also use wget

curl -X PUT https://PEM-SERVER-IP/pem/api/v16/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
}

r = requests.put(https://PEM-SERVER-IP/pem/api/v16/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": [
      {
```

(continues on next page)

(continued from previous page)

```
        "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
};

fetch(https://PEM-SERVER-IP/pem/api/v16/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}`

- Update the email group information.*

Body parameter

```
{
  "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",
```

(continues on next page)

(continued from previous page)


```
        "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
      }
    ]
  }
}
```


Parameters

Name	In	Type	Required	Description
email_group_id	path	string	true	none
X-Auth-Token	header	string	true	Token of authorization
body	body	*EmailGroupsPutData*	true	none

Example responses

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Excluded databases added successfully	None

Response Schema

Note

This operation does not require authentication

delete_*email_groups*{email_group_id}

Code samples

```
#  You can also use wget

curl -X DELETE https://PEM-SERVER-IP/pem/api/v16/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/v16/email/groups/{email_group_id},␣
↪headers = headers)

print(r.json())
```

```
const headers = {
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v16/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}`

- Delete the email group.*

Parameters

Name	In	Type	Required	Description
email_group_id	path	string	true	none
X-Auth-Token	header	string	true	Token of authorization

Example responses

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Email group deleted successfully	None

Response Schema

> **Note**
>
> This operation does not require authentication

Include/Exclude Database Monitoring

get_*server_exclude_database*{sid}

Code samples

```
#  You can also use wget

curl -X GET https://PEM-SERVER-IP/pem/api/v16/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/v16/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/v16/server/exclude_database/{sid},
{
  method: GET,

  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

`GET /server/exclude_database/{sid}`

- Retrieve information for the excluded databases, identified by the server id.*

Parameters

Name	In	Type	Required	Description
sid	path	string	true	Server ID
X-Auth-Token	header	string	true	Token of authorization

Example responses

```
{
  "server_id": 0,
  "name": "string",
  "exclude_databases": [
    "string"
  ]
}
```

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Status 200	*ExcludeDatabases*

Note

This operation does not require authentication

put_*server_exclude_database*{sid}

Code samples

```
#  You can also use wget

curl -X PUT https://PEM-SERVER-IP/pem/api/v16/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/v16/server/exclude_database/{sid},␣
```

(continues on next page)

(continued from previous page)

```
↪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/v16/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}`

- Update the excluded database information.*

Body parameter

```
{
  "exclude_databases": [
    "string"
  ]
}
```

Parameters

Name	In	Type	Required	Description
sid	path	string	true	Server ID
X-Auth-Token	header	string	true	Token of authorization
body	body	*Exclude-Databas-esPutData*	true	none

Example responses

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Email group updated successfully	None

Response Schema

Note

This operation does not require authentication

delete_*server_exclude_database*{sid}

Code samples

```
#  You can also use wget

curl -X DELETE https://PEM-SERVER-IP/pem/api/v16/server/exclude_database/{sid} \
  -H X-Auth-Token: string
```

```
import requests
headers = {
  X-Auth-Token: string
}

r = requests.delete(https://PEM-SERVER-IP/pem/api/v16/server/exclude_database/{sid},␣
↪headers = headers)

print(r.json())
```

```
const headers = {
  X-Auth-Token:string
};

fetch(https://PEM-SERVER-IP/pem/api/v16/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}`

- Delete the excluded databases.*

Parameters

Name	In	Type	Required	Description
sid	path	string	true	Server ID
X-Auth-Token	header	string	true	Token of authorization

Example responses

Responses

Status	Meaning	Description	Schema
200	*Invoking the server installer from the command line*	Excluded databases deleted successfully	None

Response Schema

> **Note**
>
> This operation does not require authentication

Schemas

Profile


```
{
  "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": [
```


(continues on next page)

(continued from previous page)


```
    {
      "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": [
```


(continues on next page)

(continued from previous page)

```
        0
      ],
      "cleared_webhook_ids": [
        0
      ]
    }
  ]
}
```

Full profile details with probe and alert configurations (returned by GET /profiles/{profile_id}).

Properties

Name	Type	Required	Restrictions	Description
id	integer	false	none	none
name	string	false	none	none
description	string	false	none	none
target_kind	string	false	none	Target type for the profile: 's' for server, 'a' for agent.
published	boolean	false	none	True when the profile is published, false if it's a draft.
parent_id	integer	false	none	Parent profile ID (null for published profiles, set for drafts).
draft_id	integer	false	none	Draft profile ID (only present for published profiles that have a draft).
target_probe_configs	[ProbeConfig]	false	none	Probe configurations associated with the profile.
target_alert_configs	[Alert]	false	none	Alert configurations associated with the profile.

Enumerated Values

Property	Value
target_kind	s
target_kind	a

ProfileDraft

```
{
  "id": 0,
  "name": "string",
  "description": "string",
  "target_kind": "s",
  "published": true,
  "target_probe_configs": [
```

(continues on next page)

(continued from previous page)


```
    {
      "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"
      ],
      "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,
```


(continues on next page)

(continued from previous page)


```
      "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
      ]
    }
  ]
}
```


Draft profile details (returned by GET /profiles/{profile_id}/draft). Does not include parent_id or draft_id.

Properties

Name	Type	Required	Restrictions	Description
id	integer	false	none	none
name	string	false	none	none
description	string	false	none	none
target_kind	string	false	none	Target type for the profile: 's' for server, 'a' for agent.
published	boolean	false	none	Always false for drafts.
target_probe_configs	[ProbeConfig]	false	none	Probe configurations associated with the draft.
target_alert_configs	[Alert]	false	none	Alert configurations associated with the draft.

Enumerated Values

Property	Value
target_kind	s
target_kind	a

ProfileSummary


```
{
  "id": 34,
  "name": "Production Agents",
  "description": "Monitoring production agents",
  "target_kind": "a",
  "active_assignments": 0,
  "published": true
}
```


A trimmed profile representation for list operations (no probe/alert config arrays).

Properties

Name	Type	Required	Restrictions	Description
id	integer	false	none	none
name	string	false	none	none
description	string	false	none	none
target_kind	string	false	none	Target type for the profile: 's' for server, 'a' for agent.
published	boolean	false	none	Convenience flag: true when status is 'published'.
active_assignments	integer	false	none	Count of active servers/agents assigned to this profile.

Enumerated Values

Property	Value
target_kind	s
target_kind	a

ProbeConfig


```
{
  "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
}
```

Properties

Name	Type	Required	Restrictions	Description
probe_id	integer	false	none	none
probe_name	string	false	none	none
enabled	boolean	false	none	none
interval	integer	false	none	Execution frequency (minutes).
lifetime	integer	false	none	History retention (days).
default_enabled	boolean	false	none	none
default_interval	integer	false	none	none
default_lifetime	integer	false	none	none
target_type	integer	false	none	Target type identifier for probe (e.g., 200=server, 100=agent).
force_enabled	boolean	false	none	none

ProbeConfigUpsert

```
{
  "probe_id": 7,
  "enabled": true,
  "enabled_by_default": true,
  "execution_frequency": 0,
  "lifetime": 0
}
```

Probe configuration fields to add or update for a profile.

Properties

Name	Type	Required	Restrictions	Description
probe_id	integer	true	none	Probe identifier to add/update.
enabled	boolean	false	none	Enable/disable this probe in the profile.
enabled_by_default	boolean	false	none	Whether the probe follows default enablement. Alias: 'use_default_enabled' accepted.
execution_frequency	integer	false	none	Probe execution frequency in minutes. Alias: 'interval' accepted.
lifetime	integer	false	none	History retention (days).

ProbeConfigsUpdateOnly


```
{
  "changed": [
    {
      "probe_id": 7,
      "enabled": true,
      "enabled_by_default": true,
      "execution_frequency": 0,
      "lifetime": 0
    }
  ]
}
```


Delta for probe configs when updating by profile id: only modify existing probes using 'changed'. Adding probes is not supported via this endpoint.

Properties

Name	Type	Required	Restrictions	Description
changed	[ProbeConfigUpsert]	false	none	Existing probe configs to modify (identified by probe_id).

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
}
```


Change an existing alert config in a profile.

Properties

Name	Type	Required	Restrictions	Description
id	integer	false	none	Alert ID to modify.
name	string	false	none	Alert name.
enabled	boolean	false	none	none
history_retention	integer	false	none	none
frequency_min	integer	false	none	none
operator	string	false	none	none
low_threshold_value	number	false	none	none
medium_threshold_value	number	false	none	none
high_threshold_value	number	false	none	none
all_alert_enable	boolean	false	none	none
email_group_id	integer	false	none	none
low_alert_enable	boolean	false	none	none
low_email_group_id	integer	false	none	none
med_alert_enable	boolean	false	none	none
med_email_group_id	integer	false	none	none
high_alert_enable	boolean	false	none	none
high_email_group_id	integer	false	none	none

Enumerated Values

Property	Value
operator	>
operator	<

AlertDeleteRef

```
{
  "id": 0
}
```

Reference to an existing alert to delete (id only).

Properties

Name	Type	Required	Restrictions	Description
id	integer	true	none	Alert config id to delete.

AlertConfigsDelta

```
{
  "changed": [
    {
      "id": 0,
      "name": "string",
```

(continues on next page)

(continued from previous page)


```
      "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
    }
  ],
  "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": [],
```


(continues on next page)

(continued from previous page)


```
      "med_webhook_ids": [],
      "high_webhook_ids": [],
      "cleared_webhook_ids": []
    }
  ],
  "deleted": [
    {
      "id": 0
    }
  ]
}
```


Delta for alert configs within a profile.

Properties

Name	Type	Required	Restrictions	Description
changed	[AlertChange]	false	none	Modify existing alerts.
added	[AlertPayload]	false	none	Add new alerts. See AlertPayload for required fields.
deleted	[AlertDeleteRef]	false	none	Delete existing alerts by id.

DeleteAndReassignPayload


```
{
  "old_profile_id": 42,
  "new_profile_id": 84
}
```


Properties

Name	Type	Required	Restrictions	Description
old_profile_id	integer	true	none	Existing profile ID to remove.
new_profile_id	integer	false	none	New profile ID to reassign associated servers/agents (optional).

ProfileAssignmentDetails


```
{
  "has_profile": true,
  "profile_id": 0,
  "profile_name": "string"
}
```

Properties

Name	Type	Required	Restrictions	Description
has_profile	boolean	false	none	none
profile_id	integer	false	none	none
profile_name	string	false	none	none

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,
  "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": []
}
```

Properties

Name	Type	Required	Restrictions	Description
alert_name	string	false	none	Name of the alert. NOTE: This property is a mandatory parameter for a POST request.
alert_template	integer	false	none	ID of the alert template. The ID must be a positive integer value. NOTE: This property is a mandatory parameter for a POST request.
description	string	false	none	none
enabled	boolean	false	none	Defines whether this alert is enabled/disabled. NOTE: This property is a mandatory parameter for a POST request.
history_retention	integer	false	none	Specifies how many days to keep the history of this alert. Possible values are from 1 to 99999. NOTE: This property is a mandatory parameter for a POST request.
frequency_min	integer	false	none	Defines how frequently (in minutes) the alert must be checked on PEM database server. Possible values are 1 to 65534. NOTE: This property is a mandatory parameter for a POST request.
operator	string	false	none	Defines the operator for checking threshold values. Valid string are ">" and "<". NOTE: This property is a mandatory parameter for the POST request.
low_threshold_value	number	false	none	Threshold value for triggering the low alert. This property can be any valid integer/float value. NOTE: It is a mandatory parameter for a POST request.

continues on next page

Table 468 – continued from previous page

Name	Type	Required	Restrictions	Description
medium_threshold_value	number	false	none	Threshold value for triggering the medium alert. This property can be any valid integer/float value. NOTE: It is a mandatory parameter for a POST request.
high_threshold_value	number	false	none	Threshold value for triggering the high alert. This property can be any valid integer/float value. NOTE: It is a mandatory parameter for a POST request.
all_alert_enable	boolean	false	none	Determines whether to send an email notification to the group specified by the parameter 'email_group_id' when the state of the alert changes.
email_group_id	integer	false	none	Defines the email group to which email notification are sent, when the state of the alert changes.
low_alert_enable	boolean	false	none	Determines whether to send an email notification to the group specified by the parameter 'low_email_group_id' when the state of the alert is set to LOW.
low_email_group_id	integer	false	none	Defines the email group to which email notifications are sent when the state of the alert changes to LOW.
med_alert_enable	boolean	false	none	Determines whether to send an email notification to the group specified by the parameter 'medium_email_group_id' when the state of the alert is set to MEDIUM.
med_email_group_id	integer	false	none	Defines the email group to which email notifications are sent when the state of the alert changes to MEDIUM.

continues on next page

Table 468 – continued from previous page

Name	Type	Required	Restrictions	Description
high_alert_enable	boolean	false	none	Determines whether to send an email notification to the group specified by the parameter 'high_email_group_id' when the state of the alert is set to HIGH.
high_email_group_id	integer	false	none	Defines the email group to which email notification are sent when the state of the alert changes to HIGH.
send_trap	boolean	false	none	Determines whether to send an SNMP trap notification when the state of the alert changes.
snmp_trap_version	integer	false	none	Defines the SNMP trap version. Supported SNMP trap versions are 1, 2 or 3.
low_send_trap	boolean	false	none	Determines whether to send SNMP trap notifications when the state of the alert changes to LOW.
med_send_trap	boolean	false	none	Determines whether to send SNMP trap notifications when the state of the alert changes to MEDIUM.
high_send_trap	boolean	false	none	Determines whether to send SNMP trap notifications when the state of the alert chagnes to HIGH.
cleared_alert_enable	boolean	false	none	Determines whether to mute the notification for 'clear alert'.
params	[AlertParams]	false	none	List of parameter/value pairs to be used by the alert template for checking alerts. NOTE: The alert template requires the all parameter to be defined.
execute_script	boolean	false	none	Determines whether to execute the provided script if the state of the alert changes to LOW/MEDIUM/HIGH.
execute_script_on_clear	boolean	false	none	Determines whether to execute the provided script if the state of the alert clears.

continues on next page

Table 468 – continued from previous page

Name	Type	Required	Restrictions	Description
execute_script_on_pem_server	boolean	false	none	Determines whether to execute the provided script on the PEM host or the host where the agent is monitoring the object.
script_code	string	false	none	The shell or batch script to be executed. Please read the documentation for detailed information.
submit_to_nagios	boolean	false	none	Defines whether to send notification to the nagios, when state of the alert is chagned.
override_default_config	boolean	false	none	Determines whether to override the default webhook configuration.
send_notification	boolean	false	none	Determines whether the webhook send notification is enabled.
low_webhook_ids	[integer]	false	none	Defines webhook IDs for which low alerts can be configured.
med_webhook_ids	[integer]	false	none	Defines webhook IDs for which medium alerts can be configured.
high_webhook_ids	[integer]	false	none	Defines webhook IDs for which high alerts can be configured.
cleared_webhook_ids	[integer]	false	none	Defines webhook IDs for which cleared alerts can be configured.

Alert


```
{
  "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": [
```


(continues on next page)

(continued from previous page)


```
    "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
  ]
}
```


Properties

Name	Type	Required	Restrictions	Description
id	integer	false	none	Alert ID. NOTE: This property is not applicable for a POST request.

continues on next page

Table 469 – continued from previous page

Name	Type	Required	Restrictions	Description
alert_name	string	false	none	Name of the alert. NOTE: This property is a mandatory parameter for a POST request.
alert_template	integer	false	none	ID of the alert template. This property must be a positive integer value. NOTE: It is a mandatory parameter for a POST request.
description	string	false	none	none
enabled	boolean	false	none	Determines whether this alert is enabled/disabled. NOTE: This property is a mandatory parameter for a POST request.
history_retention	integer	false	none	Specifies how many days to keep the history of this alert. Possible values are from 1 to 99999. NOTE: This property is a mandatory parameter for a POST request.
frequency_min	integer	false	none	Defines in minutes how frequently the alert must be checked on a PEM database server. Possible values are from 1 to 65534. NOTE: This property is a mandatory parameter for a POST request.
auto_created	boolean	false	none	Specifies whether the alert is auto-created.
template_id	string	false	none	Defines the alert template.
params	[AlertParams]	false	none	List of parameter/value pairs to be used by the alert template for checking alerts. NOTE: The alert template requires the all parameter to be defined.
param_names	[string]	false	none	Defines the name of parameters.
param_types	string	false	none	Defines the type of parameters. Example: INTEGER/STRING..

continues on next page

Table 469 – continued from previous page

Name	Type	Required	Restrictions	Description
params_units	[string]	false	none	Defines the unit used for parameters.Example: 'DAYS','True or False','Hours','Minutes' etc
threshold_unit	string	false	none	Units used to calculate the threshold, such as MB/kB/seconds/days/... This value is only used for the GUI.
operator	string	false	none	Defines the operator for checking threshold values. Valid strings are ">" and "<". NOTE: This property is a mandatory parameter for a POST request.
low_threshold_value	number	false	none	Threshold value for triggering the low alert. This property can be any valid integer/float value. NOTE: It is a mandatory parameter for the POST request.
medium_threshold_value	number	false	none	Threshold value for triggering the medium alert. This property can be any valid integer/float value. NOTE: It is a mandatory parameter for the POST request.
high_threshold_value	number	false	none	Threshold value for triggering the high alert. It can be any valid integer/float value. NOTE: It is a mandatory parameter for the POST request.
send_email	boolean	false	none	Determines whether to send an email when the state of the alert changes.
all_alert_enable	boolean	false	none	Determines whether to send an email notification to the group specified by the parameter 'email_group_id' when the state of the alert changes.
email_group_id	integer	false	none	Defines the email group to which email notification are sent when the state of the alert changes.

continues on next page

Table 469 – continued from previous page

Name	Type	Required	Restrictions	Description
low_alert_enable	boolean	false	none	Determines whether to send an email notification to the group specified by the parameter 'low_email_group_id' when the state of the alert is set to LOW.
low_email_group_id	integer	false	none	Defines the email group to which email notification are sent when the state of the alert changes to LOW.
med_alert_enable	boolean	false	none	Determines whether to send an email notification to the group specified by the parameter 'medium_email_group_id' when the state of the alert is set to MEDIUM.
med_email_group_id	integer	false	none	Defines the email group to which email notification are sent when state of the alert changes to MEDIUM.
high_alert_enable	boolean	false	none	Determines whether to send an email notification to the group specified by the parameter 'high_email_group_id' when the state of the alert is set to HIGH.
high_email_group_id	integer	false	none	Defines the email group to which email notification are sent when the state of the alert changes to HIGH.
send_trap	boolean	false	none	Determines whether to send SNMP trap notifications when the state of the alert changes.
snmp_trap_version	integer	false	none	Defines the SNMP trap version. Supported SNMP trap versions are 1, 2 or 3
low_send_trap	boolean	false	none	Determines whether to send SNMP trap notifications when the state of the alert changes to LOW.
med_send_trap	boolean	false	none	Determines whether to send SNMP trap notification when the state of the alert changes to MEDIUM.

continues on next page

Table 469 – continued from previous page

Name	Type	Required	Restrictions	Description
high_send_trap	boolean	false	none	Determines whether to send SNMP trap notification when the state of the alert changes to HIGH.
cleared_alert_enable	boolean	false	none	Determines whether to mute the notification for 'clear alert'.
execute_script	boolean	false	none	Determines whether to execute the provided script if the state of the alert changes to LOW/MEDIUM/HIGH.
execute_script_on_clear	boolean	false	none	Determines whether to execute the provided script if the state of the alert is cleared.
execute_script_on_pem_server	boolean	false	none	Determines whether to execute the provided script on the PEM host or the host where the agent is monitoring the object.
script_code	string	false	none	The shell or batch script to be executed. Please read the documentation for detailed information.
submit_to_nagios	boolean	false	none	Determines whether to send notification to the nagios when the state of the alert changes.
override_default_config	boolean	false	none	Determines whether to override the default webhook configuration.
send_notification	boolean	false	none	Determines whether to enable the webhook send notification.
low_webhook_ids	[integer]	false	none	Webhook IDs for which low alerts are configured
med_webhook_ids	[integer]	false	none	Webhook IDs for which medium alerts are configured
high_webhook_ids	[integer]	false	none	Webhook IDs for which high alerts are configured
cleared_webhook_ids	[integer]	false	none	Webhook IDs for which cleared alerts are configured

AlertParams

```
{
  "paramname": "Exclude tables smaller than",
  "paramvalue": "string"
}
```

Properties

Name	Type	Required	Restrictions	Description
paramname	string	false	none	Name of the parameter. NOTE: This property must present on the alert template parameter list.Example:
paramvalue	string	false	none	Value for the given parameter

Enumerated Values

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"
}
```

Properties

Name	Type	Required	Restrictions	Description
param_names	string	false	none	Defines the name of parameters.
param_types	string	false	none	Defines the type of parameters. Example: INTEGER/STRING..

continues on next page

Table 472 – continued from previous page

Name	Type	Required	Restrictions	Description
params_units	string	false	none	Defines the unit used for parameters.Example: 'DAYS','True or False','Hours','Minutes' etc

ProbeDependencyParams

```
{
  "internal_name": "string"
}
```

Properties

Name	Type	Required	Restrictions	Description
internal_name	string	false	none	Internal name of the probe.

AlternateServerProbeCode

```
{
  "server_version_id": 0,
  "server_probe_code": "string",
  "extension_version": 0
}
```

Properties

Name	Type	Required	Restrictions	Description
server_version_id	integer	false	none	Server version ID for the monitored server
server_probe_code	string	false	none	SQL query to run on the specified server version
extension_version	integer	false	none	Specifies the extension version. Valid for extension target type

DeletedAlternateServerProbeCode

```
{
  "server_version_id": 0
}
```

Properties

Name	Type	Required	Restrictions	Description
server_version_id	integer	false	none	Server version ID for the monitored server

ProbeColumn

```
{
  "pc_id": 0,
  "pc_unit": "string",
  "pc_graphable": true
}
```

Properties

Name	Type	Required	Restrictions	Description
pc_id	integer	false	none	Column identifier
pc_unit	string	false	none	Returns the unit of the column.
pc_graphable	boolean	false	none	Determines whether the data of the column is graphable on a chart.

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,
```

(continues on next page)

(continued from previous page)


```
  "espan": 0,
  "line_span": 0,
  "sel_metrics_L": [
    {}
  ],
  "sel_metrics_T": [
    {}
  ],
  "sel_metrics_C": [
    {}
  ],
  "shared": [
    0
  ],
  "shared_all": true
}
```


Properties

Name	Type	Required	Restrictions	Description
chart_title	string	false	none	Chart title. NOTE: This property is a mandatory parameter for a POST request.
chart_type	string	false	none	Chart catagory. NOTE: This property is a mandatory parameter for a POST request.
chart_category	string	false	none	Chart type. NOTE: This property is a mandatory parameter for a POST request.
chart_description	string	false	none	Description of the chart. NOTE: This property is a mandatory parameter for a POST request.
reference_id	string	false	none	Internal identifier for the chart.
chart_level	integer	false	none	Defines the chart level. NOTE: This property is a mandatory parameter for a POST request.
chart_line_ext	[integer]	false	none	Defines the line chart extrapolation. NOTE: This property is a mandatory parameter for a POST request.

continues on next page

Table 477 – continued from previous page

Name	Type	Required	Restrictions	Description
chart_line_ext_metric	string	false	none	Defines the line chart extrapolation metrics. NOTE: This property is a mandatory parameter for a POST request.
chart_line_ext_metric_options	[integer]	false	none	Defines the line chart extrapolation metrics option. NOTE: This property is a mandatory parameter for a POST request.
chart_line_ext_opt	string	false	none	Defines the line chart extrapolation options. NOTE: This property is a mandatory parameter for a POST request.
chart_line_ext_val	string	false	none	Defines the line chart extrapolation values. NOTE: This property is a mandatory parameter for a POST request.
chart_line_extrapolated_type	string	false	none	Defines the line chart extrapolation type. NOTE: This property is a mandatory parameter for a POST request.
chart_line_points	integer	false	none	Defines the line chart extrapolation point.
chart_line_span	[integer]	false	none	Defines the line chart span. NOTE: This property is a mandatory parameter for a POST request.
chart_refresh	integer	false	none	Chart refresh time
espan	integer	false	none	Chart span
line_span	integer	false	none	Chart line span
sel_metrics_L	[object]	false	none	Defines the line chart metrics and its probes data.
sel_metrics_T	[object]	false	none	Defines the table chart metrics and its probes data.
sel_metrics_C	[object]	false	none	Defines the capacity manager chart metrics and its probes data.
shared	[integer]	false	none	Defines the role ID to which chart was shared.
shared_all	boolean	false	none	Determines whether the chart needs to be shared with everyone.

DashboardExportTemplate

```
{
  "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
}
```

Properties

Name	Type	Required	Restrictions	Description
name	string	false	none	Defines the dashboard name. NOTE: This property is a mandatory parameter for a POST request.
descp	string	false	none	Provides the dashboard description.
reference_id	string	false	none	Internal identifier for the dashboard
font	string	false	none	Defines the dashboard font.
font_size	integer	false	none	Defines the dashboard font size.
level	integer	false	none	Defines the dashboard level.
is_ops	boolean	false	none	Determines whether the dashboard is for ops.
show_title	boolean	false	none	Determines whether the dashboard title is shown.
design_layout	[object]	false	none	Defines the dashboard design layout data.
c_charts	[object]	false	none	Defines the dependant custom charts and its probes data.

continues on next page

Table 478 – continued from previous page

Name	Type	Required	Restrictions	Description
shared	[integer]	false	none	Defines the role ID to which the dashboard is shared.
shared_all	boolean	false	none	Determines whether the dashboard must be shared with everyone

CustomDashboardExportData

```
{
  "dashboards": [
    0
  ]
}
```

Properties

Name	Type	Required	Restrictions	Description
dashboards	[integer]	false	none	List of the dashboard IDs

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
    }
  ]
}
```

Properties

Name	Type	Required	Restrictions	Description
version	integer	false	none	Version of the dashboard export
dashboards	[DashboardExportTemplate]	false	none	none

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
    }
  ]
}
```

Properties

Name	Type	Required	Restrictions	Description
version	integer	false	none	Version of the dashboard export
skip_overwrite	boolean	false	none	Determines whether the existing chart can be skipped.

continues on next page

Table 481 – continued from previous page

Name	Type	Required	Restrictions	Description
skip_overwrite_probe	boolean	false	none	Determines whether the existing dependant probes can be skipped.
skip_overwrite_chart	boolean	false	none	Determines whether the existing dependant charts can be skipped.
dashboards	[DashboardExportTemplate]	false	none	none

CustomChartExportData

```
{
  "charts": [
    0
  ]
}
```

Properties

Name	Type	Required	Restrictions	Description
charts	[integer]	false	none	List of the chart IDs

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": [
```

(continues on next page)

(continued from previous page)

```
        0
      ],
      "chart_refresh": 0,
      "espan": 0,
      "line_span": 0,
      "sel_metrics_L": [
        {}
      ],
      "sel_metrics_T": [
        {}
      ],
      "sel_metrics_C": [
        {}
      ],
      "shared": [
        0
      ],
      "shared_all": true
    }
  ]
}
```

Properties

Name	Type	Required	Restrictions	Description
version	integer	false	none	Version of the chart export
charts	[ChartExport-Template]	false	none	none

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
```

(continues on next page)

(continued from previous page)


```
      ],
      "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
    }
  ]
}
```


Properties

Name	Type	Required	Restrictions	Description
version	integer	false	none	Version of the chart export
skip_overwrite	boolean	false	none	Determines whether the existing chart can be skipped.
skip_overwrite_probe	boolean	false	none	Determines whether existing dependant probes can be skipped.
charts	[ChartExport-Template]	false	none	none

AlertImportExportTemplate


```
{
  "name": "string",
  "description": "string",
  "reference_id": "string",
  "default_history_retention": 0,
  "object_type": 0,
```


(continues on next page)

(continued from previous page)

```
  "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
        }
      ],
      "internal_name": "string"
    }
  ]
}
```

Properties

Name	Type	Required	Restrictions	Description
name	string	false	none	Name of the alert template. NOTE: This property is a mandatory parameter for a POST request.
description	string	false	none	Description of the alert template. NOTE: This property is a mandatory parameter for a POST request.
reference_id	string	false	none	Internal identifier for the alert template
default_history_retention	integer	false	none	Defines the history retention period in days for this type of alerts. NOTE: This property is a mandatory parameter for a POST request.

continues on next page

Table 485 – continued from previous page

Name	Type	Required	Restrictions	Description
object_type	integer	false	none	Defines the type of the object to which this alert template applies. - A system-wide alert template is 50. - An alert template applicable to an agent is 100. - An alert template applicable to a managed server is 200. - An alert template applicable to a database for a particular monitored server is 300. - An alert template applicable to a schema on a particular database for the managed server is 400. - An alert template applicable to a table on a particular schema for the database on a monitored server is 500. - An alert template applicable to an index on a particular schema for the database on a monitored server is 600. - An alert template applicable to a sequence on a particular schema for the database on a monitored server is 700. - An alert template applicable to a function on a particular schema for the database on a monitored server is 800. NOTE: This property is a mandatory parameter for a POST request.

continues on next page

Table 485 – continued from previous page

Name	Type	Required	Restrictions	Description
sql	string	false	none	SQL query, possibly parameterized, to get a numeric value indicating the threshold reached. NOTE: This property is a mandatory parameter for a POST request.
applicable_on_server	string	false	none	Specifies the type of monitored server to which the template applies. Possible values are ALL, POSTGRES_SERVER, ADVANCED_SERVER. NOTE: This property is a mandatory parameter for a POST request.
default_check_frequency	integer	false	none	Specifies how frequently (in minutes) to check for the alert that this template defines. NOTE: This property is a mandatory parameter for a POST request.
threshold_unit	string	false	none	Units used to calculate the threshold, such as MB/kB/seconds/days/... This value is only used for the GUI.
is_system_template	boolean	false	none	Determines whether the given template is system-defined or user-defined. NOTE: This property is not applicable for POST/PUT requests.
is_auto_create	boolean	false	none	Determines whether an alert should be created as soon as an agent or server is added to PEM..
thresholds	[number]	false	none	Default threshold values. Values in the thresholds array must be in a sorted order based on the operator.
operator	string	false	none	Operator to compare threshold values
high_threshold_value	string	false	none	High threshold value
medium_threshold_value	string	false	none	Medium threshold value
low_threshold_value	string	false	none	Low threshold value

continues on next page

Table 485 – continued from previous page

Name	Type	Required	Restrictions	Description
info_sql	string	false	none	Defines the SQL query that provides detailed information about the alert on the dashboard.
probes	[CustomProbeImportPostData]	false	none	none

Enumerated Values

Property	Value
applicable_on_server	ALL
applicable_on_server	POSTGRES_SERVER
applicable_on_server	ADVANCED_SERVER
operator	>
operator	<

CustomAlertTemplateExportData

```
{
  "alert_templates": [
    0
  ]
}
```

Properties

Name	Type	Required	Restrictions	Description
alert_templates	[integer]	false	none	List of alert template IDs

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",
```

(continues on next page)

(continued from previous page)


```
      "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"
        }
      ]
    }
  ]
}
```

Properties

Name	Type	Required	Restrictions	Description
version	integer	false	none	Version of the alert template export
alert_templates	[AlertImportExportTemplate]	false	none	none

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",
      "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": [
```

(continues on next page)

(continued from previous page)

```
            {
              "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"
        }
      ]
    }
  ]
}
```

Properties

Name	Type	Required	Restrictions	Description
version	integer	false	none	Version of the alert template export
skip_overwrite	boolean	false	none	Determines whether the existing alert template can be skipped.
skip_overwrite_probe	boolean	false	none	Determines whether existing dependent probes can be skipped.
alert_templates	[AlertImportExportTemplate]	false	none	none

AlertTemplate

```
{
  "name": "string",
  "description": "string",
  "object_type": 0,
  "param_names": [
    "string"
  ],
  "param_types": "string",
```

(continues on next page)

(continued from previous page)

```
  "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"
}
```

Properties

Name	Type	Required	Restrictions	Description
name	string	false	none	Name of the alert template. NOTE: This property is a mandatory parameter for a POST request.
description	string	false	none	Description of the alert template. NOTE: This property is a mandatory parameter for a POST request.

continues on next page

Table 490 – continued from previous page

Name	Type	Required	Restrictions	Description
object_type	integer	false	none	Defines the type of the object to which this alert template applies. - A system-wide alert template is 50. - An alert template applicable to an agent is 100. - An alert template applicable to a managed server is 200. - An alert template applicable to a database for a particular monitored server is 300. - An alert template applicable to a schema on a particular database for the managed server is 400. - An alert template applicable to a table on a particular schema for the database on a monitored server is 500. - An alert template applicable to an index on a particular schema for the database on a monitored server is 600. - An alert template applicable to a sequence on a particular schema for the database on a monitored server is 700. - An alert template applicable to a function on a particular schema for the database on a monitored server is 800. NOTE: This property is a mandatory parameter for a POST request.
param_names	[string]	false	none	Defines the name of parameters.

continues on next page

Table 490 – continued from previous page

Name	Type	Required	Restrictions	Description
param_types	string	false	none	Defines the type of parameters. Example: INTEGER/STRING..
params_units	[string]	false	none	Defines the unit used for parameters.Example: 'DAYS','True or False','Hours','Minutes' etc
sql	string	false	none	SQL query, possibly parameterized, to get a numeric value indicating threshold reached. NOTE: This property is a mandatory parameter for a POST request.
applicable_on_server	string	false	none	Specifies the type of monitored server to which the template applies. Possible values are ALL, POSTGRES_SERVER, and ADVANCED_SERVER. NOTE: This property is a mandatory parameter for a POST request.
default_check_frequency	integer	false	none	Defines how frequently (in minutes) to check for the alert defined in this template. NOTE: This property is a mandatory parameter for a POST request.
default_history_retention	integer	false	none	Defines the history retention period in days for this type of alerts. NOTE: This property is a mandatory parameter for a POST request.
probe_dependency_list	[string]	false	none	Defines the list of probes on which alert template is dependent.
threshold_unit	string	false	none	Units used to calculate the threshold, such as MB/kB/seconds/days/... This value is only used for the GUI.

continues on next page

Table 490 – continued from previous page

Name	Type	Required	Restrictions	Description
is_system_template	boolean	false	none	Determines whether the given template is system-defined or user-defined. NOTE: This property is not applicable for POST/PUT requests.
is_auto_create	boolean	false	none	Determines whether an alert should be created as soon as an agent or server is added to PEM.Note: Only supported for Agent and Server level i.e object_type 100 and 200 respectively.
operator	string	false	none	Operator to compare threshold values.Only supported for Agent and Server level i.e object_type 100 and 200 respectively.
low_threshold_value	number	false	none	Low threshold value.Only supported for Agent and Server level i.e object_type 100 and 200 respectively.
medium_threshold_value	number	false	none	Medium threshold value.Only supported for Agent and Server level i.e object_type 100 and 200 respectively.
high_threshold_value	number	false	none	High threshold value.Only supported for Agent and Server level i.e object_type 100 and 200 respectively.
info_sql	string	false	none	Specifies the SQL query that provides detailed information about the alert on the dashboard.

Enumerated Values

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,
  "high_threshold_value": 0,
  "info_sql": "string"
}
```

Properties

Name	Type	Required	Restrictions	Description
name	string	false	none	Name of the alert template. NOTE: This property is a mandatory parameter for a POST request.
description	string	false	none	Description of the alert template. NOTE: This property is a mandatory parameter for a POST request.

continues on next page

Table 492 – continued from previous page

Name	Type	Required	Restrictions	Description
object_type	integer	false	none	Defines the type of the object to which this alert template applies. - A system-wide alert template is 50. - An alert template applicable to an agent is 100. - An alert template applicable to a managed server is 200. - An alert template applicable to a database for a particular monitored server is 300. - An alert template applicable to a schema on a particular database for the managed server is 400. - An alert template applicable to a table on a particular schema for the database on a monitored server is 500. - An alert template applicable to an index on a particular schema for the database on a monitored server is 600. - An alert template applicable to a sequence on a particular schema for the database on a monitored server is 700. - An alert template applicable to a function on a particular schema for the database on a monitored server is 800. NOTE: This property is a mandatory parameter for a POST request.

continues on next page

Table 492 – continued from previous page

Name	Type	Required	Restrictions	Description
params	[AlertTemplateParams]	false	none	List of parameter/value pairs to be used by the alert template for checking alerts. NOTE: The alert template requires the all parameter to be defined.
sql	string	false	none	SQL query, possibly parameterized, to get a numeric value indicating threshold reached. NOTE: This property is a mandatory parameter for a POST request.
applicable_on_server	string	false	none	Specifies the type of monitored server to which the template applies. Possible values are ALL, POSTGRES_SERVER, and ADVANCED_SERVER. NOTE: This property is a mandatory parameter for a POST request.
default_check_frequency	integer	false	none	Defines how frequently (in minutes) to check for the alert defined in this template. NOTE: This property is a mandatory parameter for a POST request.
default_history_retention	integer	false	none	Defines the history retention period in days for this type of alerts. NOTE: This property is a mandatory parameter for a POST request.
probe_dependency_list	[ProbeDependencyParams]	false	none	Defines the list of probes on which alert template is dependent.
threshold_unit	string	false	none	Units used to calculate the threshold, such as MB/kB/seconds/days/... This value is only used for the GUI.
is_system_template	boolean	false	none	Determines whether the given template is system-defined or user-defined. NOTE: This property is not applicable for POST/PUT requests.

continues on next page

Table 492 – continued from previous page

Name	Type	Required	Restrictions	Description
is_auto_create	boolean	false	none	Determines whether an alert should be created as soon as an agent or server is added to PEM.Note: Only supported for Agent and Server level i.e object_type 100 and 200 respectively.
operator	string	false	none	Operator to compare threshold values.Only supported for Agent and Server level i.e object_type 100 and 200 respectively.
low_threshold_value	number	false	none	Low threshold value.Only supported for Agent and Server level i.e object_type 100 and 200 respectively.
medium_threshold_value	number	false	none	Medium threshold value.Only supported for Agent and Server level i.e object_type 100 and 200 respectively.
high_threshold_value	number	false	none	High threshold value.Only supported for Agent and Server level i.e object_type 100 and 200 respectively.
info_sql	string	false	none	Specifies the SQL query that provides detailed information about the alert on the dashboard.

Enumerated Values

Property	Value
applicable_on_server	ALL
applicable_on_server	POSTGRES_SERVER
applicable_on_server	ADVANCED_SERVER
operator	>
operator	<

AlertTemplatePut

```
{
  "name": "string",
  "description": "string",
```

(continues on next page)

(continued from previous page)

```
  "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"
}
```

Properties

Name	Type	Required	Restrictions	Description
name	string	false	none	Name of the alert template. NOTE: This property is a mandatory parameter for a POST request.
description	string	false	none	Description of the alert template. NOTE: This property is a mandatory parameter for a POST request.
sql	string	false	none	SQL query, possibly parameterized, to get a numeric value indicating threshold reached. NOTE: This property is a mandatory parameter for a POST request.
applicable_on_server	string	false	none	Specifies the type of monitored server to which the template applies. Possible values are ALL, POSTGRES_SERVER, and ADVANCED_SERVER. NOTE: This property is a mandatory parameter for a POST request.

continues on next page

Table 494 – continued from previous page

Name	Type	Required	Restrictions	Description
default_check_frequency	integer	false	none	Defines how frequently (in minutes) to check for the alert defined in this template. NOTE: This property is a mandatory parameter for a POST request.
default_history_retention	integer	false	none	Defines the history retention period in days for this type of alerts. NOTE: This property is a mandatory parameter for a POST request.
probe_dependency_list	object	false	none	Defines the list of probes on which alert template is dependent.
» deleted	[ProbeDependencyParams]	false	none	List of parameter/value pairs to be deleted.
» added	[ProbeDependencyParams]	false	none	List of parameter/value pairs to be added.
threshold_unit	string	false	none	Units used to calculate the threshold, such as MB/kB/seconds/days/... This value is only used for the GUI.
is_system_template	boolean	false	none	Determines whether the given template is system-defined or user-defined. NOTE: This property is not applicable for POST/PUT requests.
info_sql	string	false	none	Specifies the SQL query that provides detailed information about the alert on the dashboard.

Enumerated Values

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
}
```


Properties

Name	Type	Required	Restrictions	Description
alert_id	integer	false	none	Alert ID
state	string	false	none	State
value	string	false	none	User-presentable value
actual_value	number	false	none	Actual value at the time of state change

Enumerated Values

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,
```


(continues on next page)

(continued from previous page)


```
  "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
      }
    ]
  }
}
```


Properties

Name	Type	Required	Restrictions	Description
id	integer	false	none	Server ID
name	string	false	none	Server name
group_id	integer	false	none	Group ID under which database server is visible to the user
group_name	string	false	none	Group name
blackout	boolean	false	none	Status of the alert blackout for the database server
status	string	false	none	Server status can be UP, DOWN, UNKNOWN, or UNMANAGED,
alerts	object	false	none	List of alerts configured for the database server, including alert status and number of alerts that are acknowledged

continues on next page

Table 498 – continued from previous page

Name	Type	Required	Restrictions	Description
» total	integer	false	none	Number of alerts configured for the database server
» acknowledged	integer	false	none	Number of acknowledged alerts
» high	integer	false	none	Number of alerts with state as 'High'
» medium	integer	false	none	Number of alerts with state as 'Medium'
» low	integer	false	none	Number of alerts with state as 'Low'
» high_acknowledged	integer	false	none	Number of acknowledged alerts with state as 'High'
» medium_acknowledged	integer	false	none	Number of acknowledged alerts with state as 'Medium'
» low_acknowledged	integer	false	none	Number of acknowledged alerts with state as 'Low'
version	string	false	none	Current version of the database server
number_connections	integer	false	none	Number of connections on the database servers
remote_monitoring	boolean	false	none	Determines whether the database server is being monitored remotely by the pemAgent
agent_id	integer	false	none	Agent ID that is monitoring the database server
sessions	object	false	none	All the session information for database servers
» last_recorded_time	number	false	none	Unix time of the last recorded time for session information
» sessions	[object]	false	none	List of all the session information for the database server
»» database_name	string	false	none	Database name
»» procpid	integer	false	none	Process ID
»» usename	string	false	none	User name
»» backend_start	number	false	none	Unix time for the session start time
»» xact_start	number	false	none	Unix time for the transaction start time
»» query_start	number	false	none	Unix time for the query start time (if any queries are running)
»» is_waiting	boolean	false	none	Is the sesion waiting?
»» is_idle	boolean	false	none	Is the session idle?
»» is_idle_in_transaction	boolean	false	none	Is the session idle in a transaction?

continues on next page

Table 498 – continued from previous page

Name	Type	Required	Restrictions	Description
»» is_vacuum	boolean	false	none	Is this a VACUUM session?
»» is_autovacuum	boolean	false	none	Is this an AUTO VACUUM session?
»» client_addr	string	false	none	Host address for the client of the session
»» client_port	integer	false	none	Client port for the session
»» memory_usage_mb	number	false	none	Memory usage (in MB) of the session (available only when monitored locally)
»» swap_usage_mb	number	false	none	Swap memory usage (in MB) of the session (available only when monitored locally)
»» cpu_usage	number	false	none	CPU Usage (in percentage) of the session (available only when monitored locally)
»» io_read_bytes	number	false	none	I/O Read (in bytes) of the session (available only when monitored locally)
»» io_write_bytes	number	false	none	I/O Write (in bytes) of the session (available only when monitored locally)
»» state	string	false	none	Current state of the session (idle, active, idle in transcation, and so forth)
»» state_change	number	false	none	Unix time of the last state change in the session

Enumerated Values

Property	Value
status	UP
status	DOWN
status	UNKNOWN
status	UNMANAGED

AgentStatus


```
{
  "id": 0,
  "name": "string",
  "group_id": 0,
  "group_name": "string",
  "blackout": true,
  "status": "UP",
  "alerts": {
```


(continues on next page)

(continued from previous page)


```
    "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
}
```


Properties

Name	Type	Required	Restrictions	Description
id	integer	false	none	Agent ID
name	string	false	none	Agent name
group_id	integer	false	none	Group ID under which agent is visible to the user
group_name	string	false	none	Group name
blackout	boolean	false	none	Determines whether an alert blackout for the agent is enabled or disabled.
status	string	false	none	Agent status can be UP, DOWN, or UNKNOWN
alerts	object	false	none	List of alerts configured for the agent, including alert status and number of alerts that are acknowledged
» total	integer	false	none	Number of alerts configured for the agent
» acknowledged	integer	false	none	Number of acknowledged alerts
» high	integer	false	none	Number of alerts with state as 'High'
» medium	integer	false	none	Number of alerts with state as 'Medium'
» low	integer	false	none	Number of alerts with state as 'Low'
» high_acknowledged	integer	false	none	Number of acknowledged alerts with state as 'High'

continues on next page

Table 500 – continued from previous page

Name	Type	Required	Restrictions	Description
» medium_acknowledged	integer	false	none	Number of acknowledged alerts with state as 'Medium'
» low_acknowledged	integer	false	none	Number of acknowledged alerts with state as 'Low'
version	string	false	none	Current version of the agent
processes	integer	false	none	Number of processes running
threads	integer	false	none	Number of threads for all the running processes
cpu_utilization	number	false	none	CPU utilization (percentage)
memory_utilization	number	false	none	Memory utilization (percentage)
swap_utilization	number	false	none	Swap memory utilization (percentage)
disk_utilization	number	false	none	Overall disk utilization (percentage)

Enumerated Values

Property	Value
status	UP
status	DOWN
status	UNKNOWN

AlertStatus


```
{
  "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"
  ],
```


(continues on next page)

(continued from previous page)

```
  "info_vals": [
    [
      "string"
    ]
  ]
}
```

Properties

Name	Type	Required	Restrictions	Description
alert_since	string	false	none	Unix time when alert state last changed
alert_id	integer	false	none	Alert internal identification number
alert_name	string	false	none	Name of the alert
alert_state	string	false	none	Current state of the alert. A 'null' value suggests alert threshold values have not currently been violated for the alert.
value	string	false	none	Current value of the alert
object_description	string	false	none	Monitoring object description/name
server_id	integer	false	none	null or ID of the server
agent_id	integer	false	none	null or ID of the agent
database	string	false	none	null or name of the database
schema	string	false	none	null or name of the schema/namespace
package	string	false	none	null or name of the EDB package
object	string	false	none	null or name of any other type of object
alert_target_level	string	false	none	Alert target level
last_processed	string	false	none	String reprenting Unix time when the alert was processed
info	string	false	none	Some information at the time when the alert was generated
info_cols	[string]	false	none	Array of column names for the information
info_vals	[array]	false	none	Detailed information associated with the alert in the form of a table (an array of an array)

Enumerated Values

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
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",
```


(continues on next page)

(continued from previous page)

```
      "extension_version": 0
    }
  ]
}
```

Properties

Name	Type	Required	Restrictions	Description
probe_id	integer	false	none	Identifier for a probe
probe_name	string	false	none	Name of the probe
enabled	boolean	false	none	Determines whether the probe is enabled or disabled by default.

continues on next page

Table 504 – continued from previous page

Name	Type	Required	Restrictions	Description
target_type	integer	false	none	Defines the type of the object to which the probe applies. - A probe with target_type 100 is responsible for collecting statistics for an agent. - A probe with target_type 200 is responsible for collecting statistics for a monitored server. - A probe with target_type 300 is responsible for collecting statistics for a database. - A probe with target_type 400 is responsible for collecting statistics for a schema. - A probe with target_type 500 is responsible for collecting statistics for a table. - A probe with target_type 600 is responsible for collecting statistics for a index. - A probe with target_type 700 is responsible for collecting statistics for a sequence. - A probe with target_type 800 is responsible for collecting statistics for a function. - A probe with target_type 900 is responsible for collecting statistics for a view. - A probe with target_type 1000 is responsible for collecting statistics for a extension.

continues on next page

Table 504 – continued from previous page

Name	Type	Required	Restrictions	Description
internal_name	string	false	none	Defines the internal name of the probe. The PEM system creates internal tables for storing probe statistics in the 'pemdata', and 'pemhistory' schema. The names of those tables are the same as the internal name.
is_system_probe	boolean	false	none	Determines whether this probe is user-defined. This flag is always false for a user-defined probe.
any_server_version	boolean	false	none	Determines whether this probe applies to any version of a Postgres variant. For more details, please consult the documentation.

continues on next page

Table 504 – continued from previous page

Name	Type	Required	Restrictions	Description
collection_method	string	false	none	Defines the method of statistics collection. PEM supports three methods for custom made probes: SQL, WMI, or batch/shell script. 'w' stands for WMI script. This type of script only applies to the Windows platform, and the target_type is 100 (for agent). 'b' stands for batch/shell script. The 'target_type' for this script is always 100 (agent). This script behaves as a batch script on Windows and a shell script on other platforms. 's' stands for SQL script. The 'target_type' is always 200 (server), 300 (database), 400 (schema), 500 (table), 600 (index), 700 (sequence), 800 (function), or 900 (view), or 1000 (extension).
interval	integer	false	none	Defines how frequently the probe collects statistics (in seconds).Must be greater than or equal to 10
lifetime	integer	false	none	Defines the history data retention time (in days). Valid range is 1-365. This value does not have any effect if 'discard_history' is set to true.
discard_history	boolean	false	none	Determines whether to discard history data. When 'discard_history' is set to false, history of the data for this probe is not stored.

continues on next page

Table 504 – continued from previous page

Name	Type	Required	Restrictions	Description
platform	string	false	none	Defines the platform where this probe can run. This value only applies to an agent-level probe.
probe_code	string	false	none	A SQL/Batch script/Shell script/WMI script to collect the statistics as per the definition of the code.
probe_columns	[object]	false	none	none
» pc_id	integer	false	none	Column identifier
» pc_name	string	false	none	Column name
» pc_data_type	string	false	none	Data type of the column. Supported datatypes are: bigint char decimal double precision integer numeric real text tex tamp timestamptz
» pc_internal_name	string	false	none	Returns the internal name of the column.
» pc_unit	string	false	none	Returns the unit of the column.
» pc_col_type	string	false	none	Returns the column as part of the primary key of the internal table, or not. 'k' stands for a key column, and 'm' stands for a member/normal column.
» pc_position	integer	false	none	Returns the position of the column in the internal table.
» pc_graphable	boolean	false	none	Determines whether the data of the column can be graphed on a chart.
» pc_calc_pit	boolean	false	none	Determines whether the server calculates the point-in-time for metric data for this column.
» pc_pit_default	boolean	false	none	Determines whether the metric is stored by point-in-time by default. Please refer to the 'Custom Probes' section in the documentation for detailed information.

continues on next page

Table 504 – continued from previous page

Name	Type	Required	Restrictions	Description
alternate_code	[Alternate-ServerProbe-Code]	false	none	none

Enumerated Values

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

CustomProbeExportData

```
{
  "probes": [
    0
  ]
}
```

Properties

Name	Type	Required	Restrictions	Description
probes	[integer]	false	none	List of the probe IDs for custom probes

CustomProbeImportData

```
{
  "version": 1,
  "skip_overwrite": true,
```

(continues on next page)

(continued from previous page)

```
  "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"
    }
  ]
}
```

Properties

Name	Type	Required	Restrictions	Description
version	integer	false	none	Version of the probe export
skip_overwrite	boolean	false	none	Determines whether existing probes can be skipped.
probes	[CustomProbeImportPostData]	false	none	none

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,
  "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"
}
```


Properties

allOf

Name	Type	Required	Restrictions	Description
anonymous	*CustomProbe-PostData*	false	none	none

and

Name	Type	Required	Restrictions	Description
anonymous	object	false	none	none

continues on next page

Table 509 – continued from previous page

Name	Type	Required	Restrictions	Description
» internal_name	string	false	none	Internal name of the probe from the system where probe was exported

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
    }
  ]
}
```

Properties

Name	Type	Required	Restrictions	Description
probe_name	string	false	none	Name of the probe

continues on next page

Table 510 – continued from previous page

Name	Type	Required	Restrictions	Description
collection_method	string	false	none	Defines the method of statistics collection. PEM supports three methods for custom made probes: SQL, WMI, or batch/shell script. 'w' stands for WMI script. This type of script only applies to the Windows platform, and the target_type is 100 (for agent). 'b' stands for batch/shell script. The 'target_type' for this script is always 100 (agent). This script behaves as a batch script on Windows and a shell script on other platforms. 's' stands for SQL script. The 'target_type' is always 200 (server), 300 (database), 400 (schema), 500 (table), 600 (index), 700 (sequence), 800 (function), 900 (view) or 1000 (extension).
enabled	boolean	false	none	Determines whether whether the probe is enabled/disabled by default.

continues on next page

Table 510 – continued from previous page

Name	Type	Required	Restrictions	Description
target_type	integer	false	none	Defines the type of the object to which the probe applies. - A probe with target_type 100 is responsible for collecting statistics for an agent. - A probe with target_type 200 is responsible for collecting statistics for a monitored server. - A probe with target_type 300 is responsible for collecting statistics for a database. - A probe with target_type 400 is responsible for collecting statistics for a schema. - A probe with target_type 500 is responsible for collecting statistics for a table. - A probe with target_type 600 is responsible for collecting statistics for a index. - A probe with target_type 700 is responsible for collecting statistics for a sequence. - A probe with target_type 800 is responsible for collecting statistics for a function. - A probe with target_type 900 is responsible for collecting statistics for a view. - A probe with target_type 1000 is responsible for collecting statistics for a extension.
any_server_version	boolean	false	none	Determines whether this probe applies to any version of a Postgres variant. For more details, please consult the documentation.

continues on next page

Table 510 – continued from previous page

Name	Type	Required	Restrictions	Description
any_extension_version	boolean	false	none	Determines whether this probe applies to any version of a Extension variant. For more details, please consult the documentation.
extension_name	string	false	none	Name of the extension. Valid names are edbspl , plpgsql , hstore , pldbgapi , edb_dblink_oci , edb_dblink_libpq , sslutils .
interval	integer	false	none	Defines how frequently the probe collects statistics (in seconds).Must be greater than or equal to 10
lifetime	integer	false	none	Defines the history data retention time (in days). Valid range is 1-365. This value does not have any effect if 'discard_history' is set to true.
discard_history	boolean	false	none	Defines whether to discard history data. When 'discard_history' is set to false, no history data is stored for this probe.
platform	string	false	none	Defines the platform where this probe can run. This value only applies to an agent-level probe.
probe_code	string	false	none	A SQL/Batch script/Shell script/WMI script to collect the statistics as per definition of the code.
probe_columns	[object]	false	none	none
» pc_name	string	false	none	Column name

continues on next page

Table 510 – continued from previous page

Name	Type	Required	Restrictions	Description
» pc_data_type	string	false	none	Data type of the column. Supported datatypes are: bigint char decimal double precision integer numeric real text tex tamp timestamptz
» pc_unit	string	false	none	Returns the unit of the column.
» pc_col_type	string	false	none	Returns the columns as part of the primary key of the internal table, or not. 'k' stands for a key column, and 'm' stands for a member/normal column.
» pc_graphable	boolean	false	none	Determines whether the data of the column can be graphed on a chart.
» pc_calc_pit	boolean	false	none	Determines whether the server calculates point-in-time for metric data for this column.
» pc_pit_default	boolean	false	none	Determines whether metric data is stored by point-in-time by default. Please refer to the 'Custom Probes' section in the documentation for detailed information.
alternate_code	[Alternate-ServerProbe-Code]	false	none	none

Enumerated Values

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

continues on next page

Table 511 – continued from previous page

Property	Value
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",
  "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
      }
    ]
  }
}
```

Properties

Name	Type	Required	Restrictions	Description
interval_min	integer	false	none	Defines how frequently the probe collects statistics (in minutes). Possible values are 0 to 2880.
interval_sec	integer	false	none	Defines how frequently the probe collects statistics (in seconds). Possible values are 0 to 59.
lifetime	integer	false	none	Defines the history data retention time (in days). Valid range is 1-365. This value does not have any effect if 'discard_history' is set to true.
enabled	boolean	false	none	Determines whether the probe is enabled by default.
probe_code	string	false	none	A SQL/Batch script/Shell script/WMI script to collect the statistics as per definition of the code.
probe_columns	object	false	none	none
» changed	[ProbeColumn]	false	none	List of the changed probe column
alternate_code	object	false	none	none
» changed	[AlternateServerProbeCode]	false	none	List of the changed alternate probe code for some server version
» deleted	[DeletedAlternateServerProbeCode]	false	none	List of the deleted alternate probe code for some server versions
» added	[AlternateServerProbeCode]	false	none	List of the alternate probe code added for some server versions

Probes

```
{
  "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,
```

(continues on next page)

(continued from previous page)

```
  "default_lifetime": 0,
  "lifetime": 0,
  "default_enabled": true,
  "enabled": true,
  "target_type_id_returned": 0
}
```

Properties

Name	Type	Required	Restrictions	Description
probe_id	integer	false	none	Unique ID of the probe
probe_name	string	false	none	Name of the probe
default_interval_min	integer	false	none	Defines the default interval for the probe to collect statistics (in minutes). Possible values are 0 to 2880.
interval_min	integer	false	none	Defines how frequently the probe collects statistics (in minutes). Possible values are 0 to 2880.
default_interval_sec	integer	false	none	Defines the default interval for the probe to collect statistics (in seconds). Possible values are 0 to 59.
interval_sec	integer	false	none	Defines how frequently the probe collects statistics (in seconds). Possible values are 0 to 59.
default_interval	integer	false	none	Defines the default interval for the probe to collect statistics (in seconds).
interval	integer	false	none	Defines how frequently the probe collects statistics (in seconds).Must be greater than or equal to 10
default_lifetime	integer	false	none	Defines the default history data retention time (in days).
lifetime	integer	false	none	Defines the history data retention time (in days). Valid range is 1-365. This value does not have any effect if 'discard_history' is set to true.
default_enabled	boolean	false	none	Defines the default probe enabled/disabled value.
enabled	boolean	false	none	Specifies whether whether the probe is enabled/disabled by default.

continues on next page

Table 513 – continued from previous page

Name	Type	Required	Restrictions	Description
target_type_id_returned	integer	false	none	Defines the return target type value. - A probe with target_type 100 is responsible for collecting statistics for an agent. - A probe with target_type 200 is responsible for collecting statistics for a monitored server. - A probe with target_type 300 is responsible for collecting statistics for a database. - A probe with target_type 400 is responsible for collecting statistics for a schema. - A probe with target_type 500 is responsible for collecting statistics for a table. - A probe with target_type 600 is responsible for collecting statistics for a index. - A probe with target_type 700 is responsible for collecting statistics for a sequence. - A probe with target_type 800 is responsible for collecting statistics for a function. - A probe with target_type 900 is responsible for collecting statistics for a view.

ProbeConfigData

```
{
  "interval_min": 0,
  "interval_sec": 0,
  "lifetime": 0,
  "enabled": true
}
```

Properties

Name	Type	Required	Restrictions	Description
interval_min	integer	false	none	Probe execution frequency (in minutes). The allowed interval value (in minutes) is 0-2880. NOTE: The total interval must be greater than 10 seconds. That is, 10 >= interval_min * 60 + interval_sec >= 172800.
interval_sec	integer	false	none	Probe execution frequency (in seconds). The allowed interval value (in seconds) is 0-59. NOTE: Total interval must be greater than 10 seconds. That is, 10 >= interval_min * 60 + interval_sec >= 172800.
lifetime	integer	false	none	Probe history data retention period (in days). That is, 1 <= lifetime <= 365.
enabled	boolean	false	none	Determines whether the probe is enabled for the given object.

CopyAgentData

```
{
  "type": "string",
  "agent_id": 0
}
```

Properties

Name	Type	Required	Restrictions	Description
type	string	false	none	Value must be "agent"
agent_id	integer	false	none	none

CopyServerData

```
{
  "type": "string",
  "server_id": 0
}
```

Properties

Name	Type	Required	Restrictions	Description
type	string	false	none	Value must be "server"
server_id	integer	false	none	none

CopyDatabaseData

```
{
  "type": "string",
  "server_id": 0,
  "database_name": "string"
}
```

Properties

Name	Type	Required	Restrictions	Description
type	string	false	none	Value must be "database"
server_id	integer	false	none	none
database_name	string	false	none	none

CopySchemaData

```
{
  "type": "string",
  "server_id": 0,
  "database_name": "string",
  "schema_name": "string"
}
```

Properties

Name	Type	Required	Restrictions	Description
type	string	false	none	Value must be "schema"
server_id	integer	false	none	none
database_name	string	false	none	none
schema_name	string	false	none	none

CopyAlertAgentData

```
{
  "type": "string",
  "agent_id": 0,
  "existing_alert_options": "I"
}
```

Properties

Name	Type	Required	Restrictions	Description
type	string	false	none	Value must be "agent"
agent_id	integer	false	none	none
existing_alert_options	string	false	none	Determines whether to ignore/Replace the duplicate alerts to be copied or to Delete all the existing alerts from target. Supported valid values are: I -> Ignore Duplicates R -> Replace Duplicates D -> Delete Existing Alerts

Enumerated Values

Property	Value
existing_alert_options	I
existing_alert_options	R
existing_alert_options	D

CopyAlertServerData

```
{
  "type": "string",
  "server_id": 0,
  "existing_alert_options": "I"
}
```

Properties

Name	Type	Required	Restrictions	Description
type	string	false	none	Value must be "server"
server_id	integer	false	none	none
existing_alert_options	string	false	none	Determines whether to ignore/Replace the duplicate alerts to be copied or to Delete all the existing alerts from target. Supported valid values are: I -> Ignore Duplicates R -> Replace Duplicates D -> Delete Existing Alerts

Enumerated Values

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"
}
```

Properties

Name	Type	Required	Restrictions	Description
type	string	false	none	Value must be "database"
server_id	integer	false	none	none
database_name	string	false	none	none
existing_alert_options	string	false	none	Determines whether to ignore/Replace the duplicate alerts to be copied or to Delete all the existing alerts from target. Supported valid values are: I -> Ignore Duplicates R -> Replace Duplicates D -> Delete Existing Alerts

Enumerated Values

Property	Value
existing_alert_options	I
existing_alert_options	R
existing_alert_options	D

CopyAlertSchemaData

```
{
  "type": "string",
  "server_id": 0,
```

(continues on next page)

(continued from previous page)

```
  "database_name": "string",
  "schema_name": "string",
  "existing_alert_options": "I"
}
```

Properties

Name	Type	Required	Restrictions	Description
type	string	false	none	Value must be "schema"
server_id	integer	false	none	none
database_name	string	false	none	none
schema_name	string	false	none	none
existing_alert_options	string	false	none	Determines whether to ignore/Replace the duplicate alerts to be copied or to Delete all the existing alerts from target. Supported valid values are: I -> Ignore Duplicates R -> Replace Duplicates D -> Delete Existing Alerts

Enumerated Values

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"
}
```

Properties

Name	Type	Required	Restrictions	Description
type	string	false	none	Value must be "table"
server_id	integer	false	none	none

continues on next page

Table 527 – continued from previous page

Name	Type	Required	Restrictions	Description
database_name	string	false	none	none
schema_name	string	false	none	none
table_name	string	false	none	none
existing_alert_options	string	false	none	Determines whether to ignore/Replace the duplicate alerts to be copied or to Delete all the existing alerts from target. Supported valid values are: I -> Ignore Duplicates R -> Replace Duplicates D -> Delete Existing Alerts

Enumerated Values

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"
}
```

Properties

Name	Type	Required	Restrictions	Description
type	string	false	none	Value must be "index"
server_id	integer	false	none	none
database_name	string	false	none	none
schema_name	string	false	none	none
index_name	string	false	none	none

continues on next page

Table 529 – continued from previous page

Name	Type	Required	Restrictions	Description
existing_alert_options	string	false	none	Determines whether to ignore/Replace the duplicate alerts to be copied or to Delete all the existing alerts from target. Supported valid values are: I -> Ignore Duplicates R -> Replace Duplicates D -> Delete Existing Alerts

Enumerated Values

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"
}
```

Properties

Name	Type	Required	Restrictions	Description
type	string	false	none	Value must be "function"
server_id	integer	false	none	none
database_name	string	false	none	none
schema_name	string	false	none	none
function_name	string	false	none	none
function_arguments	string	false	none	none

continues on next page

Table 531 – continued from previous page

Name	Type	Required	Restrictions	Description
existing_alert_options	string	false	none	Determines whether to ignore/Replace the duplicate alerts to be copied or to Delete all the existing alerts from target. Supported valid values are: I -> Ignore Duplicates R -> Replace Duplicates D -> Delete Existing Alerts

Enumerated Values

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"
}
```

Properties

Name	Type	Required	Restrictions	Description
type	string	false	none	Value must be "sequence"
server_id	integer	false	none	none
database_name	string	false	none	none
schema_name	string	false	none	none
sequence_name	string	false	none	none
existing_alert_options	string	false	none	Determines whether to ignore/Replace the duplicate alerts to be copied or to Delete all the existing alerts from target. Supported valid values are: I -> Ignore Duplicates R -> Replace Duplicates D -> Delete Existing Alerts

Enumerated Values

Property	Value
existing_alert_options	I
existing_alert_options	R
existing_alert_options	D

Agent

```
{
  "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"
  ]
}
```

Returns the list of objects, which contains agent information, version, capabilities, and details about the host on which the agent has been installed.

For example: \

```
[{
  /* 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,
```

(continues on next page)

(continued from previous page)

```
  /* 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"
}]
```

a

Properties

Name	Type	Required	Restrictions	Description
id	integer	false	none	Identity of the pem-agent
name	string	false	none	Description (name) of the agent.
version	string	false	none	Version of the pem-agent
profile_id	integer	false	none	Determine whether this agent has to be managed by the specified profile id.
active	boolean	false	none	Determines whether the agent is active. An active agent is responsible for collecting the statistics for the host and the monitored server.
heartbeat_tolerance	integer	false	none	heartbeat tolerance (in seconds) for the agent. Specifies how late a heartbeat can be before it is considered missed.
alert_blackout	boolean	false	none	Determines whether the alert is blacked out for the agent.
owner	string	false	none	Username of user who installed the agent. This user is called the owner.
team	string	false	none	Name of the role/user who can see this agent. A null/empty value indicates all users can see this agent.
gid	integer	false	none	Identity of the group
tags	[string]	false	none	List of tags associated with the agent
ignore_mnt_points	[string]	false	none	Used to ignore the mount point for disk usage calculation

Agents

```
[
  {
    "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"
    ]
  }
]
```

Returns the list of objects, which contains agent information, version, capabilities, and details about the host on which the agent has been installed.

For example: \

```
[{
  /* 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
↪",
```

(continues on next page)

(continued from previous page)

```
  /* Monitored servers */
  "server_bind": "Postgres Enterprise Manager Server, PostgreSQL 11, PostgreSQL 12"
}]
```

a

Properties

Name	Type	Required	Restrictions	Description
id	integer	true	none	Identity of the pem-agent
name	string	true	none	Description (name) of the agent.
version	string	true	none	Version of the pem-agent
profile_id	integer	false	none	Determine whether this agent has to be managed by the specified profile id.
active	boolean	true	none	Determines whether the agent is active. An active agent is responsible for collecting the statistics for the host and the monitored server.
heartbeat_tolerance	integer	true	none	heartbeat tolerance (in seconds) for the agent. Specifies how late a heartbeat can be before it is considered missed.
alert_blackout	boolean	true	none	Determines whether the alert is blacked out for the agent.
owner	string	true	none	Username of user who installed the agent. This user is called the owner.
team	string	true	none	Name of the role/user who can see this agent. A null/empty value indicates all users can see this agent.
gid	integer	false	none	Identity of the group
tags	[string]	false	none	List of tags associated with the agent
ignore_mnt_points	[string]	false	none	Used to ignore the mount point for disk usage calculation

Server

```
{
  "id": 0,
  "name": "string",
```

(continues on next page)

(continued from previous page)


```
  "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"
  ],
  "agent_description": "string",
  "sslcompression": false,
  "use_ssh_tunnel": false,
  "tunnel_host": "string",
  "tunnel_port": 22,
  "tunnel_username": "string",
  "tunnel_authentication": false,
```


(continues on next page)

(continued from previous page)

```
  "tunnel_identity_file": "string",
  "tunnel_keep_alive": 0,
  "post_connection_sql": "string"
}
```

Properties

Name	Type	Required	Restrictions	Description
id	integer	false	none	Server ID.NOTE: This property is not applicable for a POST request.
name	string	false	none	Server name
host	string	false	none	Server host
port	integer	false	none	Server port
db	string	false	none	Maintenance databse
profile_id	integer	false	none	Profile ID
serviceid	string	false	none	Server service ID
connection_params	[object]	false	none	List of connection parameters for the server.
» name	string	false	none	Parameter name
» keyword	string	false	none	Parameter keyword
» value	any	false	none	Parameter value
active	boolean	false	none	Determines whether the server is active.
alert_blackout	boolean	false	none	Determines whether the alerts on the server are blacked out.
owner	integer	false	none	ID of the owner who registered the server with PEM.
team	string	false	none	Defines the role/username who can see server information.
server_owner	string	false	none	Name of the user who registered the server on the PEM database server.
is_remote_monitoring	boolean	false	none	Determines whether this server is being monitored remotely (that is, the monitoring pemAgent does not reside on the same host as the monitored server).
tags	[string]	false	none	List of tags associated with the server.
replication_solution	string	false	none	Replication solution used by the server
efm_cluster_name	string	false	none	Cluster name of EFM
efm_service_name	string	false	none	Service name of EFM
efm_installation_path	string	false	none	Installation path of EFM
patroni_cluster_name	string	false	none	Cluster name of Patroni
patroni_installation_path	string	false	none	Installation path of Patroni

continues on next page

Table 537 – continued from previous page

Name	Type	Required	Restrictions	Description
patroni_config_path	string	false	none	Configuration path of Patroni
comment	string	false	none	Description of the server
username	string	false	none	Username for connecting the database server
gid	integer	false	none	ID of the server group
server_group_name	string	false	none	Name of the server group
db_res	string	false	none	Database restriction SQL clause
role	string	false	none	The role the connection will use
is_edb	boolean	false	none	Is server an EDB Advanced Server?
agent_id	integer	false	none	Agent ID bound to the server
asb_host	string	false	none	PEM database host used by agent while creating PEM connection
asb_port	integer	false	none	PEM database port used by agent while creating PEM connection
asb_username	string	false	none	PEM database user name used by agent while creating PEM connection
asb_database	string	false	none	PEM database to which agent will connect
asb_sslmode	string	false	none	PEM database SSL mode
agent_allowtakeover	boolean	false	none	Specifies whether the server can be taken over by another agent.
agent_capability_list	[string]	false	none	Array of agent capabilities
agent_description	string	false	none	Agent description
sslcompression	boolean	false	none	Determines whether data sent over SSL should be compressed.
use_ssh_tunnel	boolean	false	none	Determines whether to use ssh tunnel.
tunnel_host	string	false	none	Tunnel host name
tunnel_port	integer	false	none	Tunnel port number
tunnel_username	string	false	none	Tunnel username
tunnel_authentication	boolean	false	none	Determines whether to use tunnel authentication.
tunnel_identity_file	string	false	none	Tunnel identity file used for authentication.
tunnel_keep_alive	integer	false	none	Interval in seconds to send keep alive packets for ssh tunnel.
post_connection_sql	string	false	none	SQL to be executed after establishing the connection.

Enumerated Values

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

ServerGroupBody

```
{
  "name": "string"
}
```

Properties

Name	Type	Required	Restrictions	Description
name	string	false	none	Server group name

ServerGroupBodyGet

```
{
  "id": 0,
  "name": "string",
  "hidden": true,
  "parent_id": 0
}
```

Properties

Name	Type	Required	Restrictions	Description
id	integer	false	none	Server group id
name	string	false	none	Server group name
hidden	boolean	false	none	Determines whether Server group is hidden
parent_id	integer	false	none	Parent server group id. Note: Cluster type node will have parent id else it will be null.

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",
  "role": "string",
  "agent_id": 0,
  "asb_host": "string",
  "asb_port": 0,
  "asb_username": "string",
```


(continues on next page)

(continued from previous page)


```
  "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"
}
```


Properties

Name	Type	Required	Restrictions	Description
name	string	false	none	Server name
host	string	false	none	Server host name or address
port	integer	false	none	Server port
db	string	false	none	Maintenance databse
profile_id	integer	false	none	Profile ID
serviceid	string	false	none	Server service ID
connection_params	[object]	false	none	List of connection parameters for the server.
» name	string	false	none	Parameter name
» keyword	string	false	none	Parameter keyword
» value	any	false	none	Parameter value
alert_blackout	boolean	false	none	Determines whether the alerts on the server are blacked out.
team	string	false	none	Defines the role/username who can see server information.
is_remote_monitoring	boolean	false	none	Determines whether this server is being monitored remotely (that is, the monitoring pemAgent does not reside on the same host as the monitored server).
tags	[object]	false	none	List of tags associated with the server.
» text	string	false	none	The name of the tag.
» color	string	false	none	The color associated with the tag.

continues on next page

Table 541 – continued from previous page

Name	Type	Required	Restrictions	Description
replication_solution	string	false	none	Replication solution used by the server
efm_cluster_name	string	false	none	Cluster name of EFM
efm_service_name	string	false	none	Service name of EFM
efm_installation_path	string	false	none	Installation path of EFM
patroni_cluster_name	string	false	none	Cluster name of Patroni
patroni_installation_path	string	false	none	Installation path of Patroni
patroni_config_path	string	false	none	Configuration path of Patroni
comment	string	false	none	Description of the server
username	string	false	none	Username for connecting the database server
gid	integer	false	none	ID of the server group
hostaddr	string	false	none	Server host address
db_res	string	false	none	Database restriction SQL clause
role	string	false	none	The role the connection will use
agent_id	integer	false	none	Agent ID bound to the server
asb_host	string	false	none	PEM database host used by agent while creating PEM connection
asb_port	integer	false	none	PEM database port used by agent while creating PEM connection
asb_username	string	false	none	PEM database user name used by agent while creating PEM connection
asb_database	string	false	none	PEM database to which agent will connect
asb_sslmode	string	false	none	PEM database SSL mode
asb_password	string	false	none	empty string
agent_allowtakeover	boolean	false	none	Specifies whether the server can be taken over by another agent.
fgcolor	string	false	none	Foreground color of server node in browser tree. Default is black(#000000)
bgcolor	string	false	none	Background color of server node in browser tree. Default is white(#FFFFFF)
sslcompression	boolean	false	none	Determines whether data sent over SSL should be compressed.
use_ssh_tunnel	boolean	false	none	Determines whether to use ssh tunnel.
tunnel_host	string	false	none	Tunnel host name
tunnel_port	integer	false	none	Tunnel port number

continues on next page

Table 541 – continued from previous page

Name	Type	Required	Restrictions	Description
tunnel_username	string	false	none	Tunnel username
tunnel_password	string	false	none	Tunnel password
tunnel_authentication	boolean	false	none	Determines whether to use tunnel authentication.
tunnel_identity_file	string	false	none	Tunnel identity file used for authentication.
tunnel_keep_alive	integer	false	none	Interval in seconds to send keep alive packets for ssh tunnel.
post_connection_sql	string	false	none	SQL to be executed after establishing the connection.

Enumerated Values

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"
```


(continues on next page)

(continued from previous page)


```
      }
    ],
    "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",
  "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",
```


(continues on next page)

(continued from previous page)

```
  "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"
}
```

Properties

Name	Type	Required	Restrictions	Description
name	string	false	none	Server name
host	string	false	none	Server host name or address
port	integer	false	none	Server port
db	string	false	none	Maintenance databse
profile_id	integer	false	none	Profile ID
serviceid	string	false	none	Server service ID
connection_params	[object]	false	none	List of connection parameters for the server.
» name	string	false	none	Parameter name
» keyword	string	false	none	Parameter keyword
» value	any	false	none	Parameter value
alert_blackout	boolean	false	none	Determines whether the alerts on the server are blacked out.
team	string	false	none	Defines the role/username who can see server information.
is_remote_monitoring	boolean	false	none	Determines whether this server is being monitored remotely (that is, the monitoring pemAgent does not reside on the same host as the monitored server).
tags	[object]	false	none	List of tags associated with the server.
» text	string	true	none	The name of the tag.
» color	string	true	none	The color associated with the tag.

continues on next page

Table 543 – continued from previous page

Name	Type	Required	Restrictions	Description
» old_text	string¦null	false	none	The previous name of the tag (used for tracking changes).
replication_solution	string	false	none	Replication solution used by the server
efm_cluster_name	string	false	none	Cluster name of EFM
efm_service_name	string	false	none	Service name of EFM
efm_installation_path	string	false	none	Installation path of EFM
patroni_cluster_name	string	false	none	Cluster name of Patroni
patroni_installation_path	string	false	none	Installation path of Patroni
patroni_config_path	string	false	none	Configuration path of Patroni
comment	string	false	none	Description of the server
username	string	false	none	Username for connecting the database server
gid	integer	false	none	ID of the server group
hostaddr	string	false	none	Server host address
db_res	string	false	none	Database restriction SQL clause
role	string	false	none	The role the connection will use
agent_id	integer	false	none	Agent ID bound to the server
asb_host	string	false	none	PEM database host used by agent while creating PEM connection
asb_port	integer	false	none	PEM database port used by agent while creating PEM connection
asb_username	string	false	none	PEM database user name used by agent while creating PEM connection
asb_database	string	false	none	PEM database to which agent will connect
asb_sslmode	string	false	none	PEM database SSL mode
asb_password	string	false	none	empty string
agent_allowtakeover	boolean	false	none	Specifies whether the server can be taken over by another agent.
fgcolor	string	false	none	Foreground color of server node in browser tree. Default is black(#000000)
bgcolor	string	false	none	Background color of server node in browser tree. Default is white(#FFFFFF)
sslcompression	boolean	false	none	Determines whether data sent over SSL should be compressed.

continues on next page

Table 543 – continued from previous page

Name	Type	Required	Restrictions	Description
use_ssh_tunnel	boolean	false	none	Determines whether to use ssh tunnel.
tunnel_host	string	false	none	Tunnel host name
tunnel_port	integer	false	none	Tunnel port number
tunnel_username	string	false	none	Tunnel username
tunnel_password	string	false	none	Tunnel password
tunnel_authentication	boolean	false	none	Determines whether to use tunnel authentication.
tunnel_identity_file	string	false	none	Tunnel identity file used for authentication.
tunnel_keep_alive	integer	false	none	Interval in seconds to send keep alive packets for ssh tunnel.
post_connection_sql	string	false	none	SQL to be executed after establishing the connection.

Enumerated Values

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": [
      {
```

(continues on next page)

(continued from previous page)

```
        "text": "tag1",
        "color": "#8789AC",
        "old_text": "tag2"
      }
    ],
    "changed": [
      {
        "old_text": "tag",
        "text": "tag3"
      }
    ]
  },
  "ignore_mnt_points": [
    "string"
  ]
}
```

Returns the list of objects, which contains agent information, version, capabilities, and details about the host on which the agent has been installed.

For example: \

```
[{
  /* 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"
}]
```

a

Properties

Name	Type	Required	Restrictions	Description
name	string	false	none	Description (name) of the agent.
heartbeat_tolerance	integer	false	none	heartbeat tolerance (in seconds) for the agent. Specifies how late a heartbeat can be before it is considered missed.
alert_blackout	boolean	false	none	Determines whether the alert is blacked out for the agent.
profile_id	integer	false	none	Determine whether this agent has to be managed by the specified profile id.
team	string	false	none	Name of the role/user who can see this agent. A null/empty value indicates all users can see this agent.
gid	integer	false	none	Identity of the group
tags	[object]	false	none	List of tags associated with the agent.
» text	string	true	none	The name of the tag.
» color	string	true	none	The color associated with the tag.
» old_text	string¦null	false	none	The previous name of the tag (used for tracking changes).
ignore_mnt_points	[string]	false	none	Used to ignore the mount point for disk usage calculation

AgentJobBodyGet

```
{
  "jobname": "string",
  "jobenabled": true,
  "jobdesc": "string"
}
```

Properties

Name	Type	Required	Restrictions	Description
jobname	string	false	none	Name of the job.
jobenabled	boolean	false	none	Determines whether the job is enabled or disabled.
jobdesc	string	false	none	Description of the job.

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": [
        true
      ],
      "jscmonthdays": [
        true
      ],
      "jscmonths": [
        true
      ],
      "jschours": [
        true
      ],
      "jscminutes": [
        true
      ],
      "jscexceptions": [
        null
      ],
      "jsctimezone": "string"
    }
  ]
}
```

Properties

Name	Type	Required	Restrictions	Description
jobid	integer	false	none	ID of the job.
jobname	string	false	none	Name of the job.
jobenabled	boolean	false	none	Determines whether the job is enabled or disabled.
jobdesc	string	false	none	Description of the job.
jobcreated	string(date-time)	false	none	Timestamp when the job was created.
jobchanged	string(date-time)	false	none	Timestamp when the job was last modified.
jlgstatus	string	false	none	Status of the last job execution.
jobnextrun	string(date-time)¦r	false	none	Timestamp of the next scheduled run.
joblastrun	string(date-time)¦r	false	none	Timestamp of the last run.
is_alert_job	boolean	false	none	Indicates if the job is an alert job.
notify	string	false	none	Notification setting for the job.
email_group_id	string	false	none	ID of the email group for notifications.
jsteps	[object]	false	none	List of steps for the job.
» jstid	integer	false	none	ID of the step.
» jstjobid	integer	false	none	ID of the job associated with the step.
» jstname	string	false	none	Name of the step.
» jstdesc	string	false	none	Description of the step.
» jstenabled	boolean	false	none	Determines whether the step is enabled or disabled.
» jstkind	boolean	false	none	Determines the kind of the step. True for SQL step, False for Batch step.
» jstcode	string	false	none	Code or script for the step.
» server_id	integer	false	none	ID of the server associated with the step.
» database_name	string	false	none	Name of the database associated with the step.
» jstonerror	string	false	none	Specifies the behavior of agent if it encounters an error while executing the step('Fail', 'Success', 'Ignore').
jschedules	[object]	false	none	List of schedules for the job.
» jscid	integer¦null	false	none	ID of the schedule.
» jscjobid	integer¦null	false	none	ID of the job associated with the schedule.
» jscname	string	false	none	Name of the schedule.
» jscdesc	string	false	none	Description of the schedule.

continues on next page

Table 547 – continued from previous page

Name	Type	Required	Restrictions	Description
» jscenabled	boolean	false	none	Determines whether the schedule is enabled or disabled.
» jscstart	string(date-time)	false	none	Start date and time of the schedule.
» jscend	string(date-time)	false	none	End date and time of the schedule.
» jscweekdays	[boolean]	false	none	List of weekdays for the schedule.
» jscmonthdays	[boolean]	false	none	List of month days for the schedule.
» jscmonths	[boolean]	false	none	List of months for the schedule.
» jschours	[boolean]	false	none	List of hours for the schedule.
» jscminutes	[boolean]	false	none	List of minutes for the schedule.
» jscexceptions	[any]	false	none	List of exceptions for the schedule.
» jsctimezone	string	false	none	Timezone of the schedule.

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"
      }
    ],
    "changed": [
      {
        "jstid": 0,
        "jstname": "string",
        "jstdesc": "string",
        "jstenabled": true,
        "jstkind": true,
        "jstcode": "string",
        "server_id": 0,
        "database_name": "string",
```

(continues on next page)

(continued from previous page)


```
        "jstonerror": "f",
        "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": "Monday",
        "jscmonthdays": 1,
        "jscmonths": "January",
        "jschours": 0,
        "jscminutes": 0,
        "jscexceptions": [
          {
            "jexdate": "2023-10-19",
            "jextime": "09:39:00"
          }
        ],
        "jsctimezone": "Asia/Calcutta"
      }
    ],
    "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": "Monday",
        "jscmonthdays": 1,
        "jscmonths": "January",
        "jschours": 0,
        "jscminutes": 0,
        "jscexceptions": [
          {
            "jexdate": "2023-10-19",
            "jextime": "09:39:00"
          }
        ],
        "jsctimezone": "Asia/Calcutta"
      }
    ]
  },
  "notify": "DEFAULT",
  "email_group_id": "string"
```


(continues on next page)

(continued from previous page)

```
}
```

Properties

Name	Type	Required	Restrictions	Description
jobname	string	false	none	Name of the job.
jobenabled	boolean	false	none	Determines whether the job is enabled or disabled.
jobdesc	string	false	none	Description of the job.
jsteps	object	false	none	none
» added	[object]	false	none	List of added steps.
»» jstname	string	false	none	Name of the step.
»» jstdesc	string	false	none	Description of the step.
»» jstenabled	boolean	false	none	Determines whether the step is enabled or disabled.
»» jstkind	boolean	false	none	Determines the kind of the step. True for SQL step, False for Batch step.
»» jstcode	string	false	none	Code or script for the step.
»» server_id	integer	false	none	ID of the server associated with the step.
»» database_name	string	false	none	Name of the database associated with the step.
»» jstonerror	string	false	none	Specifies the behavior of agent if it encounters an error while executing the step('Fail', 'Success', 'Ignore').
» changed	[object]	false	none	List of changed steps.
»» jstid	integer	false	none	ID of the step.
»» jstname	string	false	none	Name of the step.
»» jstdesc	string	false	none	Description of the step.
»» jstenabled	boolean	false	none	Determines whether the step is enabled or disabled.
»» jstkind	boolean	false	none	Determines the kind of the step. True for SQL step, False for Batch step.
»» jstcode	string	false	none	Code or script for the step.
»» server_id	integer	false	none	ID of the server associated with the step.
»» database_name	string	false	none	Name of the database associated with the step.
»» jstonerror	string	false	none	Specifies the behavior of agent if it encounters an error while executing the step('Fail', 'Success', 'Ignore').
»» jstnextrun	string	false	none	none
jschedules	object	false	none	none
» added	[object]	false	none	List of added schedules.

continues on next page

Table 548 – continued from previous page

Name	Type	Required	Restrictions	Description
»» jscname	string	false	none	Name of the schedule.
»» jscdesc	string	false	none	Description of the schedule.
»» jscenabled	boolean	false	none	Determines whether the schedule is enabled or disabled.
»» jscstart	string	false	none	Start date and time of the schedule(YYYY-MM-DD HH:mm ±HH:mm).
»» jscend	string	false	none	End date and time of the schedule(YYYY-MM-DD HH:mm ±HH:mm).
»» jscweekdays	[any]	false	none	List of weekdays for the schedule.
»» jscmonthdays	[any]	false	none	List of month days for the schedule.Note: Here,'32' represents Last Day of the Month
»» jscmonths	[any]	false	none	List of months for the schedule.
»» jschours	[any]	false	none	List of hours(0-23) for the schedule.
»» jscminutes	[any]	false	none	List of minutes(0-59) for the schedule.
»» jscexceptions	[object]	false	none	List of exceptions for the schedule.
»»» jexdate	string(date)	false	none	Date of the schedule exception.
»»» jextime	string(time)	false	none	Time of the schedule exception.
»» jsctimezone	string	false	none	Timezone of the schedule.
» changed	[object]	false	none	List of changed schedules.
»» jscid	integer¦null	false	none	ID of the schedule.
»» jscname	string	false	none	Name of the schedule.
»» jscdesc	string	false	none	Description of the schedule.
»» jscenabled	boolean	false	none	Determines whether the schedule is enabled or disabled.
»» jscstart	string	false	none	Start date and time of the schedule(YYYY-MM-DD HH:mm ±HH:mm).
»» jscend	string	false	none	End date and time of the schedule(YYYY-MM-DD HH:mm ±HH:mm).
»» jscweekdays	[any]	false	none	List of weekdays for the schedule.

continues on next page

Table 548 – continued from previous page

Name	Type	Required	Restrictions	Description
»» jscmonthdays	[any]	false	none	List of month days for the schedule.Note: Here,'32' represents Last Day of the Month
»» jscmonths	[any]	false	none	List of months for the schedule.
»» jschours	[any]	false	none	List of hours(0-23) for the schedule.
»» jscminutes	[any]	false	none	List of minutes(0-59) for the schedule.
»» jscexceptions	[object]	false	none	List of exceptions for the schedule.
»»» jexdate	string(date)	false	none	Date of the schedule exception.
»»» jextime	string(time)	false	none	Time of the schedule exception.
»» jsctimezone	string	false	none	Timezone of the schedule.
notify	string	false	none	Notification setting for the job.
email_group_id	string	false	none	ID of the email group for notifications.

Enumerated Values

Property	Value
jstonerror	f
jstonerror	s
jstonerror	i
jstonerror	f
jstonerror	s
jstonerror	i
jscweekdays	Monday
jscweekdays	Tuesday
jscweekdays	Wednesday
jscweekdays	Thursday
jscweekdays	Friday
jscweekdays	Saturday
jscweekdays	Sunday
jscmonthdays	1
jscmonthdays	2
jscmonthdays	3
jscmonthdays	4
jscmonthdays	5
jscmonthdays	6
jscmonthdays	7
jscmonthdays	8
jscmonthdays	9
jscmonthdays	10

continues on next page

Table 549 – continued from previous page

Property	Value
jscmonthdays	11
jscmonthdays	12
jscmonthdays	13
jscmonthdays	14
jscmonthdays	15
jscmonthdays	16
jscmonthdays	17
jscmonthdays	18
jscmonthdays	19
jscmonthdays	20
jscmonthdays	21
jscmonthdays	22
jscmonthdays	23
jscmonthdays	24
jscmonthdays	25
jscmonthdays	26
jscmonthdays	27
jscmonthdays	28
jscmonthdays	29
jscmonthdays	30
jscmonthdays	31
jscmonthdays	32
jscmonths	January
jscmonths	February
jscmonths	March
jscmonths	April
jscmonths	May
jscmonths	June
jscmonths	July
jscmonths	August
jscmonths	September
jscmonths	October
jscmonths	November
jscmonths	December
jschours	0
jschours	1
jschours	2
jschours	3
jschours	4
jschours	5
jschours	6
jschours	7
jschours	8
jschours	9
jschours	10
jschours	22
jscminutes	0
jscminutes	1
jscminutes	2
jscminutes	3

continues on next page

Table 549 – continued from previous page

Property	Value
jscminutes	4
jscminutes	5
jscminutes	6
jscminutes	7
jscminutes	8
jscminutes	9
jscminutes	10
jscminutes	31
jscminutes	32
jscminutes	33
jscminutes	34
jscminutes	35
jscminutes	36
jscminutes	37
jscminutes	38
jscminutes	39
jscminutes	51
jscminutes	52
jscminutes	53
jscminutes	54
jscminutes	55
jscminutes	56
jscminutes	57
jscminutes	58
jscminutes	59
jsctimezone	Asia/Calcutta
jsctimezone	Canada/Pacific
jsctimezone	Asia/Tokyo
jsctimezone	etc
jscweekdays	Monday
jscweekdays	Tuesday
jscweekdays	Wednesday
jscweekdays	Thursday
jscweekdays	Friday
jscweekdays	Saturday
jscweekdays	Sunday
jscmonthdays	1
jscmonthdays	2
jscmonthdays	3
jscmonthdays	4
jscmonthdays	5
jscmonthdays	6
jscmonthdays	7
jscmonthdays	8
jscmonthdays	9
jscmonthdays	10
jscmonthdays	11
jscmonthdays	12
jscmonthdays	13
jscmonthdays	14

continues on next page

Table 549 – continued from previous page

Property	Value
jscmonthdays	15
jscmonthdays	16
jscmonthdays	17
jscmonthdays	18
jscmonthdays	19
jscmonthdays	20
jscmonthdays	21
jscmonthdays	22
jscmonthdays	23
jscmonthdays	24
jscmonthdays	25
jscmonthdays	26
jscmonthdays	27
jscmonthdays	28
jscmonthdays	29
jscmonthdays	30
jscmonthdays	31
jscmonthdays	32
jscmonths	January
jscmonths	February
jscmonths	March
jscmonths	April
jscmonths	May
jscmonths	June
jscmonths	July
jscmonths	August
jscmonths	September
jscmonths	October
jscmonths	November
jscmonths	December
jschours	0
jschours	1
jschours	2
jschours	3
jschours	4
jschours	5
jschours	6
jschours	7
jschours	8
jschours	9
jschours	10
jschours	22
jscminutes	0
jscminutes	1
jscminutes	2
jscminutes	3
jscminutes	4
jscminutes	5
jscminutes	6
jscminutes	7

continues on next page

Table 549 – continued from previous page

Property	Value
jscminutes	8
jscminutes	9
jscminutes	10
jscminutes	31
jscminutes	32
jscminutes	33
jscminutes	34
jscminutes	35
jscminutes	36
jscminutes	37
jscminutes	38
jscminutes	39
jscminutes	51
jscminutes	52
jscminutes	53
jscminutes	54
jscminutes	55
jscminutes	56
jscminutes	57
jscminutes	58
jscminutes	59
jsctimezone	Asia/Calcutta
jsctimezone	Canada/Pacific
jsctimezone	Asia/Tokyo
jsctimezone	etc
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": "Monday",
      "jscmonthdays": 1,
      "jscmonths": "January",
      "jschours": 0,
      "jscminutes": 0,
      "jscexceptions": [
```

(continues on next page)

(continued from previous page)

```
        {
          "jexdate": "2023-10-19",
          "jextime": "09:39:00"
        }
      ],
      "jsctimezone": "Asia/Calcutta"
    }
  ],
  "notify": "DEFAULT",
  "email_group_id": "string"
}
```

Properties

Name	Type	Required	Restrictions	Description
jobname	string	false	none	Name of the job.
jobenabled	boolean	false	none	Determines whether the job is enabled or disabled.
jobdesc	string	false	none	Description of the job.
jsteps	[object]	false	none	List of steps for the job.
» jstname	string	false	none	Name of the step.
» jstdesc	string	false	none	Description of the step.
» jstenabled	boolean	false	none	Determines whether the step is enabled or disabled.
» jstkind	boolean	false	none	Determines the kind of the step. True for SQL step, False for Batch step.
» jstcode	string	false	none	Code or script for the step.
» server_id	integer	false	none	ID of the server associated with the step.
» database_name	string	false	none	Name of the database associated with the step.
» jstonerror	string	false	none	Specifies the behavior of agent if it encounters an error while executing the step('Fail', 'Success', 'Ignore').
jschedules	[object]	false	none	List of schedules for the job.
» jscname	string	false	none	Name of the schedule.
» jscdesc	string	false	none	Description of the schedule.
» jscenabled	boolean	false	none	Determines whether the schedule is enabled or disabled.
» jscstart	string	false	none	Start date and time of the schedule(YYYY-MM-DD HH:mm ±HH:mm).

continues on next page

Table 550 – continued from previous page

Name	Type	Required	Restrictions	Description
» jscend	string	false	none	End date and time of the schedule(YYYY-MM-DD HH:mm ±HH:mm).
» jscweekdays	[any]	false	none	List of weekdays for the schedule.
» jscmonthdays	[any]	false	none	List of month days for the schedule.Note: Here,'32' represents Last Day of the Month
» jscmonths	[any]	false	none	List of months for the schedule.
» jschours	[any]	false	none	List of hours(0-23) for the schedule.
» jscminutes	[any]	false	none	List of minutes(0-59) for the schedule.
» jscexceptions	[object]	false	none	List of exceptions for the schedule.
»» jexdate	string(date)	false	none	Date of the schedule exception.
»» jextime	string(time)	false	none	Time of the schedule exception.
» jsctimezone	string	false	none	Timezone of the schedule.
notify	string	false	none	Notification setting for the job.
email_group_id	string	false	none	ID of the email group for notifications.

Enumerated Values

Property	Value
jstonerror	f
jstonerror	s
jstonerror	i
jscweekdays	Monday
jscweekdays	Tuesday
jscweekdays	Wednesday
jscweekdays	Thursday
jscweekdays	Friday
jscweekdays	Saturday
jscweekdays	Sunday
jscmonthdays	1
jscmonthdays	2
jscmonthdays	3
jscmonthdays	4
jscmonthdays	5
jscmonthdays	6
jscmonthdays	7
jscmonthdays	8

continues on next page

Table 551 – continued from previous page

Property	Value
jscmonthdays	9
jscmonthdays	10
jscmonthdays	11
jscmonthdays	12
jscmonthdays	13
jscmonthdays	14
jscmonthdays	15
jscmonthdays	16
jscmonthdays	17
jscmonthdays	18
jscmonthdays	19
jscmonthdays	20
jscmonthdays	21
jscmonthdays	22
jscmonthdays	23
jscmonthdays	24
jscmonthdays	25
jscmonthdays	26
jscmonthdays	27
jscmonthdays	28
jscmonthdays	29
jscmonthdays	30
jscmonthdays	31
jscmonthdays	32
jscmonths	January
jscmonths	February
jscmonths	March
jscmonths	April
jscmonths	May
jscmonths	June
jscmonths	July
jscmonths	August
jscmonths	September
jscmonths	October
jscmonths	November
jscmonths	December
jschours	0
jschours	1
jschours	2
jschours	3
jschours	4
jschours	5
jschours	6
jschours	7
jschours	8
jschours	9
jschours	10
jschours	22
jscminutes	0
jscminutes	1

continues on next page

Table 551 – continued from previous page

Property	Value
jscminutes	2
jscminutes	3
jscminutes	4
jscminutes	5
jscminutes	6
jscminutes	7
jscminutes	8
jscminutes	9
jscminutes	10
jscminutes	31
jscminutes	32
jscminutes	33
jscminutes	34
jscminutes	35
jscminutes	36
jscminutes	37
jscminutes	38
jscminutes	39
jscminutes	51
jscminutes	52
jscminutes	53
jscminutes	54
jscminutes	55
jscminutes	56
jscminutes	57
jscminutes	58
jscminutes	59
jsctimezone	Asia/Calcutta
jsctimezone	Canada/Pacific
jsctimezone	Asia/Tokyo
jsctimezone	etc
notify	DEFAULT
notify	ON_FAILURE
notify	ALWAYS
notify	NEVER

Database


```
{
  "database_name": "string",
  "connections_allowed": true,
  "system_database": true,
  "recorded_time": "2019-08-24T14:15:22Z",
  "encoding": "string",
  "server_id": 0
}
```

Properties

Name	Type	Required	Restrictions	Description
database_name	string	false	none	Database name
connections_allowed	boolean	false	none	Determines whether a connection to the database is allowed.
system_database	boolean	false	none	Determines whether the database is a system database.
recorded_time	string(date-time)	false	none	Specifies when the database was first detected by the PEM agent.
encoding	string	false	none	Database encoding
server_id	integer	false	none	Server ID of database

Schema

```
{
  "schema_name": "string",
  "database_name": "string",
  "server_id": 0,
  "recorded_time": "2019-08-24T14:15:22Z"
}
```

Properties

Name	Type	Required	Restrictions	Description
schema_name	string	false	none	Schema name
database_name	string	false	none	Database name of schema
server_id	integer	false	none	Server ID schema
recorded_time	string(date-time)	false	none	Specifies when schema was first detected by PEM agent.

Table

```
{
  "table_name": "string",
  "schema_name": "string",
  "database_name": "string",
  "server_id": 0,
  "has_primary_key": true,
  "recorded_time": "string"
}
```

Properties

Name	Type	Required	Restrictions	Description
table_name	string	false	none	Table name
schema_name	string	false	none	Schema name
database_name	string	false	none	Database name
server_id	integer	false	none	Server ID
has_primary_key	boolean	false	none	Determines whether table has a primary key.
recorded_time	string	false	none	Specifies when table was first detected by the PEM agent.

Index

```
{
  "index_name": "string",
  "ind_keys": [
    0
  ],
  "table_name": "string",
  "schema_name": "string",
  "database_name": "string",
  "server_id": 0,
  "recorded_time": "string"
}
```

Properties

Name	Type	Required	Restrictions	Description
index_name	string	false	none	Index name
ind_keys	[integer]	false	none	Indexed column positions in table
table_name	string	false	none	Table name
schema_name	string	false	none	Schema
database_name	string	false	none	Database name
server_id	integer	false	none	Server ID
recorded_time	string	false	none	Specifies when index was first detected by PEM agent.

Sequence

```
{
  "sequence_name": "string",
  "schema_name": "string",
  "database_name": "string",
  "server_id": 0,
```

(continues on next page)

(continued from previous page)

```
  "recorded_time": "2019-08-24T14:15:22Z"
}
```

Properties

Name	Type	Required	Restrictions	Description
sequence_name	string	false	none	Sequence name
schema_name	string	false	none	Schema name
database_name	string	false	none	Database name
server_id	integer	false	none	Server ID
recorded_time	string(date-time)	false	none	Specifies when sequence was first detected by PEM agent.

Function

```
{
  "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"
}
```

Properties

Name	Type	Required	Restrictions	Description
function_name	string	false	none	Function name
function_type	string	false	none	Function type
return_type	string	false	none	Function return type
arg_types	[string]	false	none	Function argument types
function_binary	string	false	none	none
extension_name	string	false	none	none
package_name	string	false	none	Function package name
schema_name	string	false	none	Function schema
database_name	string	false	none	Function database name
server_id	integer	false	none	Function server id

continues on next page

Table 557 – continued from previous page

Name	Type	Required	Restrictions	Description
recorded_time	string(date-time)	false	none	Specifies when function was first detected by PEM agent.

View

```
{
  "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"
}
```

Properties

Name	Type	Required	Restrictions	Description
view_name	string	false	none	View name
view_type	string	false	none	View type
ispopulated	boolean	false	none	Determines whether the view is populated.
view_owner	string	false	none	View owner
definition	string	false	none	View definition
tablespace_name	string	false	none	View tablespace
schema_name	string	false	none	View schema
database_name	string	false	none	View database
server_id	integer	false	none	View server ID
recorded_time	string	false	none	Specifies when view was first detected by PEM agent.

Webhook

```
{
  "id": 0,
  "name": "string",
  "url": "string",
  "enabled": true,
  "method": "POST",
  "http_headers": [
    {
      "http_header_id": 0,
```

(continues on next page)

(continued from previous page)

```
      "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"
}
```

Properties

Name	Type	Required	Restrictions	Description
id	integer	false	none	Webhook ID
name	string	false	none	Webhook name
url	string	false	none	Webhook URL
enabled	boolean	false	none	Determines whether the webhook is enabled.
method	string	false	none	Request method
http_headers	*Query information*]	false	none	List of the HTTP header key/value pairs to be used by the webhook
payload_template	string	false	none	Payload template. Below is payload example for slack.
low_alert	boolean	false	none	Determines whether the low alert is enabled.
med_alert	boolean	false	none	Determines whether the medium alert is enabled.
high_alert	boolean	false	none	Determines whether the high alert is enabled.
cleared_alert	boolean	false	none	Determines whether the cleared alert is enabled.
payload_type	string	false	none	Payload type

Enumerated Values

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"
    }
  ],
  "payload_template": "{ Alert Name:%AlertName%,type:Webhook }",
  "low_alert": true,
  "med_alert": true,
  "high_alert": true,
  "cleared_alert": true,
  "payload_type": "ALERT"
}
```

Properties

Name	Type	Required	Restrictions	Description
name	string	false	none	Webhook name
url	string	false	none	Webhook URL
enabled	boolean	false	none	Determines whether the webhook is enabled.
method	string	false	none	Request method
http_headers	*Query information*]	false	none	List of the HTTP header key/value pairs to be used by the webhook
payload_template	string	false	none	Payload template
low_alert	boolean	false	none	Determines whether the low alert is enabled.
med_alert	boolean	false	none	Determines whether the medium alert is enabled.
high_alert	boolean	false	none	Determines whether the high alert is enabled.
cleared_alert	boolean	false	none	Determines whether the cleared alert is enabled.
payload_type	string	false	none	Payload type

Enumerated Values

Property	Value
method	POST
method	PUT
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
}
```

Properties

Name	Type	Required	Restrictions	Description
name	string	false	none	Webhook name
url	string	false	none	Webhook URL
enabled	boolean	false	none	Determines whether the webhook is enabled.

continues on next page

Table 563 – continued from previous page

Name	Type	Required	Restrictions	Description
method	string	false	none	Request method
http_headers	object	false	none	none
» added	*Query information*]	false	none	List of the HTTP header key/value pairs to be added
» changed	*Query information*]	false	none	List of the HTTP header key/value pairs to be updated
» deleted	[object]	false	none	List of the HTTP header IDs to be deleted
»» http_header_id	integer	false	none	ID of the HTTP header
» payload_type	string	false	none	Payload type
payload_template	string	false	none	Payload template. Below is payload example for slack.
low_alert	boolean	false	none	Determines whether the low alert is enabled.
med_alert	boolean	false	none	Determines whether the medium alert is enabled.
high_alert	boolean	false	none	Determines whether the high alert is enabled.
cleared_alert	boolean	false	none	Determines whether the cleared alert is enabled.

Enumerated Values

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"
}
```

Properties

Name	Type	Required	Restrictions	Description
http_header_id	integer	false	none	ID of the HTTP header

continues on next page

Table 565 – continued from previous page

Name	Type	Required	Restrictions	Description
http_header_key	string	false	none	Name of the HTTP header key
http_header_value	string	false	none	Name of the HTTP header key

WebhookHTTPHeaders

```
{
  "http_header_key": "string",
  "http_header_value": "string"
}
```

Properties

Name	Type	Required	Restrictions	Description
http_header_key	string	false	none	Name of the HTTP header key
http_header_value	string	false	none	Name of the HTTP header key

EmailGroups

```
{
  "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"
}
```

Properties

Name	Type	Required	Restrictions	Description
id	integer	false	none	Email group ID
oid	integer	false	none	Email group options ID
gid	integer	false	none	Email group ID
name	string	false	none	Email group name

continues on next page

Table 567 – continued from previous page

Name	Type	Required	Restrictions	Description
to_addr	string	false	none	Email group options receiver address
cc_addr	string	false	none	Email group options CC address
bcc_addr	string	false	none	Email group options BCC address
from_addr	string	false	none	Email group options sender address
reply_to_addr	string	false	none	Email group options reply to address
subject_prefix	string	false	none	Email group options subject prefix
from_time	string	false	none	Email group options from time to select time range for members
to_time	string	false	none	Email group options to time to select time range for members

EmailGroupsPostData

```
{
  "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"
    }
  ]
}
```

Properties

Name	Type	Required	Restrictions	Description
name	string	false	none	Email group name
options	[EmailGroupOptions]	false	none	List of the email group options to be used by the email group

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",
        "subject_prefix": "string",
        "from_time": "00:00:00",
        "to_time": "23:59:59"
      }
    ],
    "deleted": [
      {
        "oid": 0
      }
    ]
  }
}
```

Properties

Name	Type	Required	Restrictions	Description
name	string	false	none	Email group name
options	object	false	none	none
» added	[Email-GroupOptions]	false	none	Email group options to be added
» changed	[Email-GroupOp-tionsChanged]	false	none	Email group options to be updated.

continues on next page

Table 569 – continued from previous page

Name	Type	Required	Restrictions	Description
» deleted	[object]	false	none	List of the HTTP header IDs to be deleted
»» oid	integer	false	none	Email group options 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"
}
```

Properties

Name	Type	Required	Restrictions	Description
to_addr	string	false	none	Email group options receiver address
cc_addr	string	false	none	Email group options CC address
bcc_addr	string	false	none	Email group options BCC address
from_addr	string	false	none	Email group options sender address
reply_to_addr	string	false	none	Email group options reply to address
subject_prefix	string	false	none	Email group options subject prefix
from_time	string	false	none	Email group options from time to select time range for members
to_time	string	false	none	Email group options to time to select time range for members

EmailGroupOptionsChanged

```
{
  "oid": 0,
  "to_addr": "string",
  "cc_addr": "string",
  "bcc_addr": "string",
  "from_addr": "string",
```

(continues on next page)

(continued from previous page)

```
  "reply_to_addr": "string",
  "subject_prefix": "string",
  "from_time": "00:00:00",
  "to_time": "23:59:59"
}
```

Properties

Name	Type	Required	Restrictions	Description
oid	integer	false	none	Email group options ID
to_addr	string	false	none	Email group options receiver address
cc_addr	string	false	none	Email group options CC address
bcc_addr	string	false	none	Email group options BCC address
from_addr	string	false	none	Email group options sender address
reply_to_addr	string	false	none	Email group options reply to address
subject_prefix	string	false	none	Email group options subject prefix
from_time	string	false	none	Email group options from time to select time range for members
to_time	string	false	none	Email group options to time to select time range for members

ExcludeDatabases

```
{
  "server_id": 0,
  "name": "string",
  "exclude_databases": [
    "string"
  ]
}
```

Properties

Name	Type	Required	Restrictions	Description
server_id	integer	false	none	Server ID
name	string	false	none	Server name
exclude_databases	[string]	false	none	Exclude database names (comma separated)

ExcludeDatabasesPutData

```
{
  "exclude_databases": [
    "string"
  ]
}
```

Properties

Name	Type	Required	Restrictions	Description
exclude_databases	[string]	false	none	Exclude database names (comma separated)

CHAPTER 37

Using the Query tool

PEM contains a feature-rich interactive development environment (IDE) that allows you to issue ad hoc SQL queries against Postgres servers.

You can access the Query tool by selecting **Tools > Query tool** or through the context menu of select nodes of the Browser tree. The Query tool allows you to:

- Issue ad hoc SQL queries.
- Execute arbitrary SQL commands.
- Edit the result set of a SELECT query if it's *Data Output tab* .
- Configure and display current connection and transaction status.
- Save the data displayed in the output panel to a CSV file.
- Review the execution plan of a SQL statement in either text, graphical, or table format (similar to https://explain.depesz.com).
- View analytical information about a SQL statement.

The Query tool features two panels:

- The upper panel displays the SQL editor. You can use the panel to enter, edit, or execute a query. It also shows the **History** tab, which you can use to view the queries that ran in the session. You can use the scratch pad to hold text snippets during editing. To reopen a closed scratch pad or open a new one, right-click the SQL editor and select **Add Panel**. You can also open new scratch pads from other panels using the same technique.
- The lower panel displays the Data Output panel. The tabbed panel displays:
- The result set returned by a query
- Information about a query's execution plan
- server messages related to the query's execution
- Any asynchronous notifications received from the server.

37.1 The Query tool toolbar

The toolbar uses context-sensitive icons that provide shortcuts to frequently performed tasks.

Icon	Behavior	Shortcut
Open File	Display a previously saved query in the SQL editor.	Accesskey + O
Save	Save a query or access the **Save** menu: - Select **Save** to save the selected content of the SQL editor panel in a file. - Select **Save As** to open a new browser dialog box and specify a new location to save the selected content of the SQL editor panel.	Accesskey + S
Save Data Changes	Save the data changes (insert, update, or delete) in the Data Output Panel to the server.	F6
Find	Search, replace, or navigate the code displayed in the SQL editor: - Select **Find** to provide a search target and search the SQL editor contents. - Select **Find next** to locate the next occurrence of the search target. - Select **Find previous** to move to the last occurrence of the search target. - Select **Pesistent find** to identify all occurrences of the search target in the editor. - Select **Replace** to locate and replace (with prompting) individual occurrences of the target. - Select **Replace all** to locate and replace all occurrences of the target in the editor. - Select **Jump** to navigate to the next occurrence of the search target.	Cmd+F Cmd+G Cmd+Shift+G Cmd+Shift+F Alt+G
Copy	Copy the content that's currently highlighted in the Data Output panel when in View/Edit data mode.	Accesskey + C
Paste	Paste a copied row into a new row when in View/Edit data mode.	Accesskey + P
Delete	Mark selected rows for deletion. Select **Save Data Changes** to update the content with your deletions.	Accesskey + D
Edit	Use options on the **Edit** menu to access text editing tools. The options operate on the text displayed in the SQL editor panel when in Query tool mode: - Select **Indent Selection** to indent the currently selected text. - Select **Unindent Selection** to remove indentation from the currently selected text. - Select **Inline Comment Selection** to enclose any lines that contain the selection in SQL-style comment notation. - Select **Inline Uncomment Selection** to remove SQL-style comment notation from the selected line. - Select **Block Comment** to enclose all lines that contain the selection in C-style comment notation. This option acts as a toggle.	Tab Shift+Tab Cmd+/ Cmd+. Shift+Cmd+/

continues on next page

Table 1 – continued from previous page

Icon	Behavior	Shortcut
Filter	Set filtering and sorting criteria for the data when in View/Edit data mode. Select the down arrow to access other filtering and sorting options: -Select **Sort/Filter** to open the sorting and filtering dialog box. - Select **Filter by Selection** to show only the rows containing the values in the selected cells. - Select **Exclude by Selection** to show only the rows that don't contain the values in the selected cells. - Select **Remove Sort/Filter** to remove any previously selected sorting or filtering options.	Accesskey + F
Limit Selector	Set a value for the maximum number of rows in a dataset.	Accesskey + R
Stop	Cancel the currently running query.	Accesskey + Q
Execute/Refresh	Execute or refresh the query selected in the SQL editor panel. Select the down arrow to access other execution options: - Select **Auto-Rollback** to roll back a transaction in case an error occurs during the transaction. - Select **Auto-Commit** to commit each transaction. Any changes made by the transaction are visible to others and durable in the event of a crash.	F5
Explain	View an explanation plan for the current query. The result of *EXPLAIN* is displayed graphically on the **Explain** tab of the output panel and in text form on the **Data Output** tab.	F7
Explain analyze	Invoke an *EXPLAIN ANALYZE* command on the current query. From the **Explain Options** menu: - Select **Verbose** to display more information about the query plan. - Select **Costs** to include information on the estimated startup and total cost of each plan node as well as the estimated number of rows and the estimated width of each row. - Select **Buffers** to include information on buffer usage. - Select **Timing** to include information about the startup time and the amount of time spent in each node of the query. - Select **Summary** to include the summary information about the query plan.	
Commit	Commit the transaction.	Shift+CTRL+M
Rollback	Roll back the transaction.	Shift+CTRL+R
Clear	Use options on the **Clear** menu to erase display contents: - Select **Clear Query Window** to erase the content of the SQL editor panel. - Select **Clear History** to erase the content of the **History** tab.	Accesskey + L
Download as CSV	Download the result set of the current query to a comma-separated list. You can specify the CSV settings through the **Preferences -> SQL editor -> CSV output** dialog box.	F8
Macros	Create, edit, or clear the macros by selecting **Manage Macros** .	

37.2 The SQL editor panel

The SQL editor panel is a workspace where you can manually provide a query, copy a query from another source, or read a query from a file. The SQL editor features syntax coloring and auto-completion.

To use auto-complete, begin typing your query. When you want the editor to suggest object names or commands that might be next in your query, press **Control+Space**. For example, type `\*SELECT \* FROM\*` (with a trailing space), and then press **Control+Space** to select from a menu of auto-complete options.

After entering a query, select **Execute/Refresh** from the toolbar. The database server receives the complete contents of the SQL editor panel to execute. To execute only a section of the code that's displayed in the SQL editor, select the text that you want the server to execute, and select **Execute/Refresh**.

The message returned by the server when a command executes is displayed on the **Messages** tab. If the command is successful, the **Messages** tab displays execution details.

The **Edit** menu helps with code formatting and commenting:

- Use auto-indent to indent text to the same depth as the previous line by pressing **Return**.
- Block indent text by selecting two or more lines and pressing **Tab**.
- Implement or remove SQL-style or toggle C-style comment notation within your code.

You can also drag certain objects from the tree to save time spent typing long object names. Text containing the object name is fully qualified with the schema name. Double quotes are added if required. For functions and procedures, the function name along with parameter names are pasted into the Query tool.

37.3 The Data Output panel

The Data Output panel displays data and statistics generated by the most recently executed query.

37.3.1 Data Output tab

The **Data Output** tab displays the result set of the query in a table format. You can:

- Select and copy from the result set.
- Use the **Execute/Refresh** options to retrieve query execution information and set query execution options.
- Select **Download as CSV** to download the content of the **Data Output** tab as a comma-delimited file.
- Edit the data in the result set of a `SELECT` query if it's updatable.

A result set is updatable if:

- All columns are either selected directly from a single table or they aren't actually a table column (for example, the concatenation of two columns). You can edit only columns that are selected directly from the table. Other columns are read-only.
- All the primary key columns or OIDs of the table are selected in the result set.

Any columns that are renamed or selected more than once are also read-only.

> **Note**
>
> To work with an updatable query result set, you must have psycopg2 driver version 2.8 or later installed.

Editable and read-only columns are identified using pencil and lock icons in the column headers.

An updatable result set is similar to the data grid in View/Edit Data mode, and you can modify it in the same way.

If auto-commit is off, data changes are made as part of the ongoing transaction. If no transaction is ongoing, a new one is started. The data changes aren't committed to the database unless the transaction is committed.

If any errors occur while saving (for example, trying to save a NULL into a column with a NOT NULL constraint), the data changes are rolled back to a savepoint to ensure any previously executed queries in the ongoing transaction aren't rolled back.

All rowsets from previous queries or commands that are displayed in the Data Output panel get discarded when you invoke another query. Open another Query tool browser tab to keep your previous results available.

37.3.2 Explain tab

To generate the Explain or Explain Analyze plan of a query, select **Explain** or **Explain Analyze** in the toolbar.

You can select more options related to **Explain** and **Explain Analyze** from the menu.

> **Note**
>
> PEM generates the Explain Analyze plan in JSON format.

On successful generation of an Explain plan, three tabs/panels appear under the **Explain** panel.

37.3.3 Graphical tab

To download the plan as an .png file, select **Download** in the top-left corner of the Explain canvas. **Download as .png** isn't supported on Internet Explorer.

The query plan that accompanies **Explain Analyze** is available on the **Data Output** tab.

37.3.4 Analysis tab

The **Analysis** tab shows the plan details in table format, generating a format similar to the one available at `explain.depsez.com`. Each row of the table represents the data for an Explain Plan node. The output can contain the node information, exclusive timing, inclusive timing, actual versus planned rows, actual rows, planned rows, or loops. Child rows of the selected row are marked with an orange dot.

If the percentage of the exclusive/inclusive timings of the total query time is:

- Greater than 90, red appears
- Greater than 50, orange appears
- Greater than 10, yellow appears

If the planner has misestimated the number of rows (actual verse planned) by:

- 10 times, yellow appears
- 100 times, orange appears
- 1000 times, red appears

37.3.5 Statistics tab

The **Statistics** tab displays information in two tables:

- Statistics per Node Type tells you how many times each node type was referenced.
- Statistics per Table tells you how many times each table was referenced by the query.

37.3.6 Messages tab

Use the **Messages** tab to view information about the most recently executed query. If the server returns an error, the error message appears on the **Messages** tab, and the syntax that caused the error is underlined in the SQL editor. If a query succeeds, the **Messages** tab shows how long the query took to complete and how many rows were retrieved.

37.3.7 Notifications tab

Use the **Notifications** tab to view the notifications using the PostgreSQL Listen/Notify feature. For more details, see the PostgreSQL documentation .

For example:

1. Execute `LISTEN "foo"` in first Query tool session.

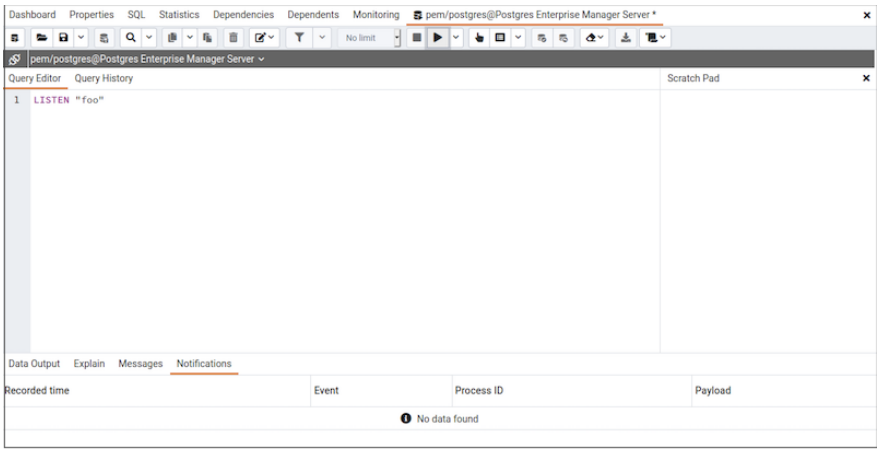


Fig. 1: Query tool - notifications listen

2. In the another Query tool session, execute the `Notify` command or `pg_notify` function to send the notification of the event together with the payload.

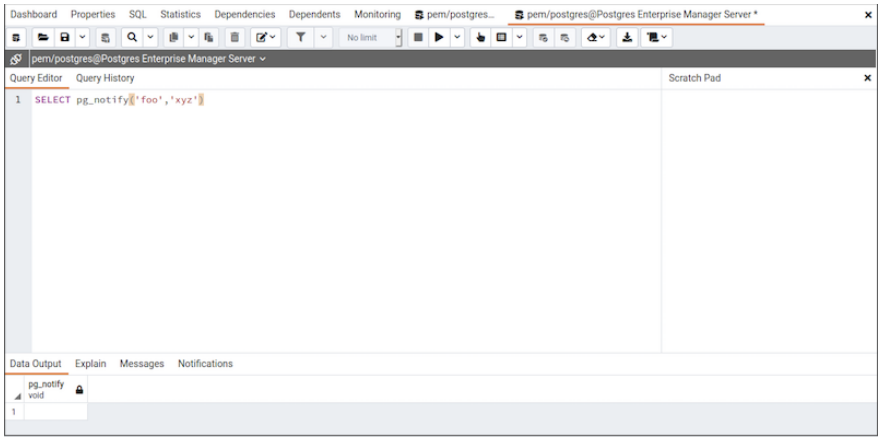


Fig. 2: Query tool - notifications notify

3. You can observe the **Notification** tab in the first Query tool session where it shows the recorded time, event, process ID, and the payload of the channel.

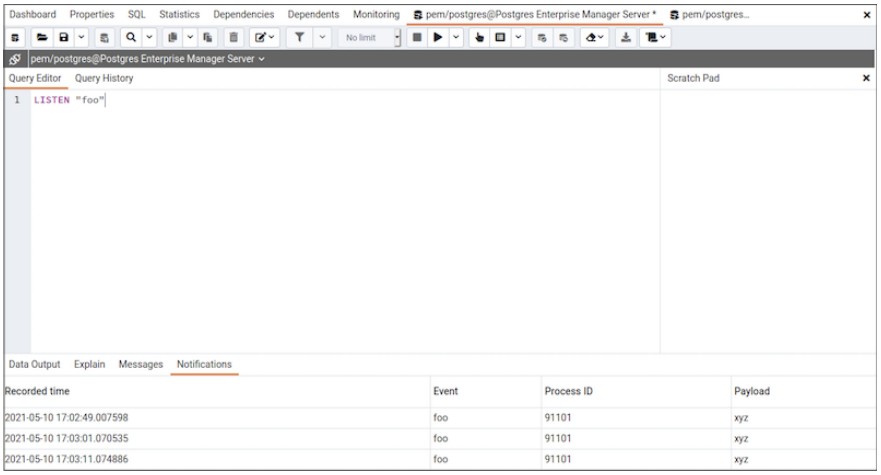


Fig. 3: Query tool - Output Notifications panel

37.4 Query History panel

Use the **Query History** tab to review activity for the current session. The **Query History** tab displays information about recent commands including:

- The date and time that a query was invoked.
- The text of the query.
- The number of rows returned by the query.
- The amount of time it took the server to process the query and return a result set.
- Messages returned by the server (not noted on the **Messages** tab).
- The source of the query (indicated by icons corresponding to the toolbar).

You can show or hide the queries generated internally by pgAdmin (during View/Edit Data or Save Data operations).

To erase the content of the **Query History** tab, select **Clear > Clear history**.

Query history is maintained across sessions for each database on a per-user basis when running in Query tool mode. In View/Edit Data mode, history isn't retained. By default, the last 20 queries are stored for each database. You can adjust this in `config_local.py` by overriding the `MAX_QUERY_HIST_STORED` value.

37.5 Connection status

Use **Connection Status** to view the current connection and transaction status by selecting the status in the Query tool.

37.6 Macros

Query tool macros enable you to execute predefined SQL queries by pressing a single key. Predefined queries can contain the placeholder `$SELECTION$` . When the macro executes, the placeholder is replaced with the currently selected text in the Query Editor pane of the Query tool.

To create a macro:

1. In the Query tool, select **Macros > Manage Macros**.
2. Select the key you want to use. Enter the name of the macro and the query.

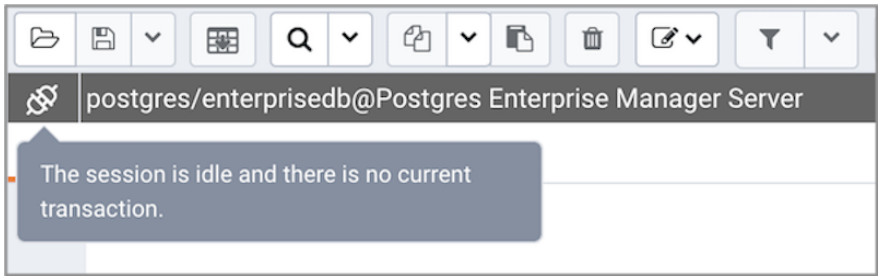


Fig. 4: Query tool - Connection Status button

3. Optionally, include the selection placeholder.
4. Select **Save**.

To clear a macro, in the Manage Macros dialog box, select the macro and select **Clear**. Respond **Yes** to the prompt.

To clear all macros, select **Clear** next to **Key**. Respond **Yes** to the prompt.

To execute a macro, select the shortcut keys, or select it from the **Macros** menu.

CHAPTER 38

Using the Schema Diff tool

Schema Diff compares schema objects between two database schemas. Use the **Tools** menu to access Schema Diff.

Schema Diff allows you to:

- Compare and synchronize the database schemas (from source to target).
- Visualize the differences between database schemas.
- List the differences in SQL statement for target schema objects.
- Generate synchronization scripts.

To open the selection panel, select **Tools > Schema Diff**. Select the source and target servers, databases, and schemas to compare. After selecting the objects, select **Compare**.

> **Note**
>
> The source and target databases must use the same major version.

You can open multiple copies of Schema Diff in individual tabs. To specify whether to open Schema Diff in a new browser tab, select the Preferences dialog box. Set **Open in new browser tab** to **true**.

The Schema Diff panel is divided into two panels: an Object Comparison panel and a DDL Comparison panel.

38.1 Schema Diff Object Comparison panel

In the Object Comparison panel, you can select the source and target servers of the same major version, databases, and schemas to compare. You can select any server listed under the browser tree whether it's connected or disconnected. If you select a server that isn't connected, then you must enter the password before using that server.

Select the databases to compare. The databases can be the same or different, and from the same or different servers.

Select the source and target schemas to compare.

After you select servers, databases, and schemas, select **Compare** to get the comparison results.

Use the lists of **Functions**, **Materialized Views**, **Tables**, **Trigger Functions**, **Procedures**, and **Views** to view the DDL statements of all the schema objects.

To filter the schema objects, select **Filter** in the upper-right corner of the Object Comparison panel. Filter the schema objects according to these criteria:

- **Identical** _ If the object is found in both schemas with the same SQL statement, then the comparison result is identical.
- **Different** _ If the object is found in both schemas with different SQL statements, then the comparison result is different.
- **Source Only** _ If the object is found only in source schema and not in target schema, then the comparison result is source only.
- **Target Only** _ If the object is found only in target schema and not in source schema, then the comparison result is target only.

Select any of the schema objects in the Object Comparison panel to display the DDL statements for that object in the DDL Comparison panel.

38.2 Schema Diff DDL Comparison panel

The DDL Comparison panel displays three columns:

- The first column displays the DDL statement of the object from the source schema.
- The second column displays the DDL statement of the object from the target schema.
- The third column displays the difference in the SQL statement of the target schema object.

To check for differences in the SQL statements, review the DDL statements of all the schema objects.

The Schema Diff tool can generate a SQL script with the differences found in the target schema object. The SQL script compares the target schema object to the SQL statement of the source schema object. To generate the script:

1. In the Object Comparison panel, select the check boxes of the schema objects.
2. Select **Generate Script**.

To open the Query tool in a new tab and display the differences in the SQL statement in the Query Editor:

1. Select the schema objects.
2. Select **Generate Script**.

If you select the schema object to check the difference generated in the DDL Comparison panel but don't select the check box of the schema object, then PEM opens the Query tool in a new tab. The Query tool displays the differences in the SQL statements in the Query Editor.

To copy the difference generated in the DDL Comparison panel, select **Copy**.

Apply the SQL Statement in the target schema to synchronize the schemas.

CHAPTER 39

Using the ERD tool

The Entity-Relationship Diagram (ERD) tool is a database design tool that provides a graphical representation of database tables, columns, and inter-relationships. ERD can give sufficient information for the database administrator to follow when developing and maintaining the database. To access the ERD tool, select **Tools > ERD Tool**. The ERD tool allows you to:

- Design and visualize the database tables and their relationships.
- Add notes to the diagram.
- Align the tables and links for a cleaner look.
- Save the diagram and open it later to continue working on it.
- Generate ready-to-run SQL from the database design.
- Generate the database diagram for an existing database.

You can open multiple copies of the ERD tool in individual tabs simultaneously. To close a copy of the ERD tool, select the **X** in the upper-right corner of the tab bar.

39.1 Toolbar

The ERD Tool toolbar uses context-sensitive icons that provide shortcuts to frequently performed tasks.

39.2 File options

Icon	Behavior	Shortcut
Open File	Load a saved diagram.	Ctrl + O
Save	Save changes to a saved diagram or save the diagram to a file.	Ctrl + S

continues on next page

Table 1 – continued from previous page

Icon	Behavior	Shortcut
Save as	Open a new dialog box and specify a new location to save the diagram.	Ctrl + Shift + S

39.3 Export options

Icon	Behavior	Shortcut
Generate SQL	Generate the DDL SQL for the diagram and open a Query tool with the generated SQL ready for execution.	Option + Ctrl + S
Download image	Save the ERD diagram in an image format.	Option + Ctrl + I

39.4 Editing options

Icon	Behavior	Shortcut
Add table	Add a new table to the diagram. In the table dialog box, enter the table details.	Option/Alt + Ctrl + A
Edit table	Edit a selected table on the diagram. In the table dialog box, enter the table details.	Option/Alt + Ctrl + E
Clone table	Clone the complete table structure, name it with a auto-generated name, and put it in the diagram.	Option/Alt + Ctrl + C
Drop table/link	Drop a selected table or link.	Option/Alt + Ctrl + D

39.5 Table relationship options

Icon	Behavior	Shortcut
1M	Open a one-to-many relationship dialog box to add a relationship between two tables. The selected table is the referencing table and has the *many* endpoint of the link.	Option/Alt + Ctrl + O
MM	Open a many-to-many relationship dialog box to add a relationship between two tables. This option creates a linked table from the selected columns of the two relating tables.	Option/Alt + Ctrl + M

39.6 Utility Options

Icon	Behavior	Shortcut
Add/Edit note	Make notes on table nodes while designing the database.	Option/Alt + Ctrl + N

continues on next page

Table 5 – continued from previous page

Icon	Behavior	Shortcut
Auto align	Align all tables and links for a cleaner look.	Option/Alt + Ctrl + L
Show details	Toggle the column details. You can also choose to show few or more column details.	Option/Alt + Shift + D

39.7 Zoom options

Icon	Behavior	Shortcut
Zoom to fit	Zoom in/out automatically and fit all the tables to the view.	Option/Alt + Shift + F
Zoom in	Zoom in on the diagram.	Option/Alt + Shift + "+"
Zoom out	Zoom out on the diagram.	Option/Alt + Shift + "-"

39.8 Table dialog

The table dialog box allows you to:

- Change the table structure details.
- Edit an existing table or add a new one.

39.9 Table node

The table node shows table details in a graphical representation:

- On the top bar:
- To show column details, select **Details toggle**.
- When you add a note, **Note** becomes enabled. Select **Note** to changes notes.
- The first row displays the table's schema name, like `public` in the example.
- The second row displays the table's name, like `users` in the example.
- The remaining rows dispalys the table's column names and their data types.
- A column that is a primary key displays as a lock key, like `id integer` in the example.
- Other columns diplay as a green column, like `email character varying` in the example.
- Select the node to drag it on the canvas.
- To open the table dialog box and change the table details, either double-click the table node or select **Edit** on the toolbar.

39.10 One-to-many link dialog box

The one-to-many link dialog box allows you to:

- Add a foregin key relationship between two tables.
- Select a Local Table that references another table. The Local Table has the `many` end point.

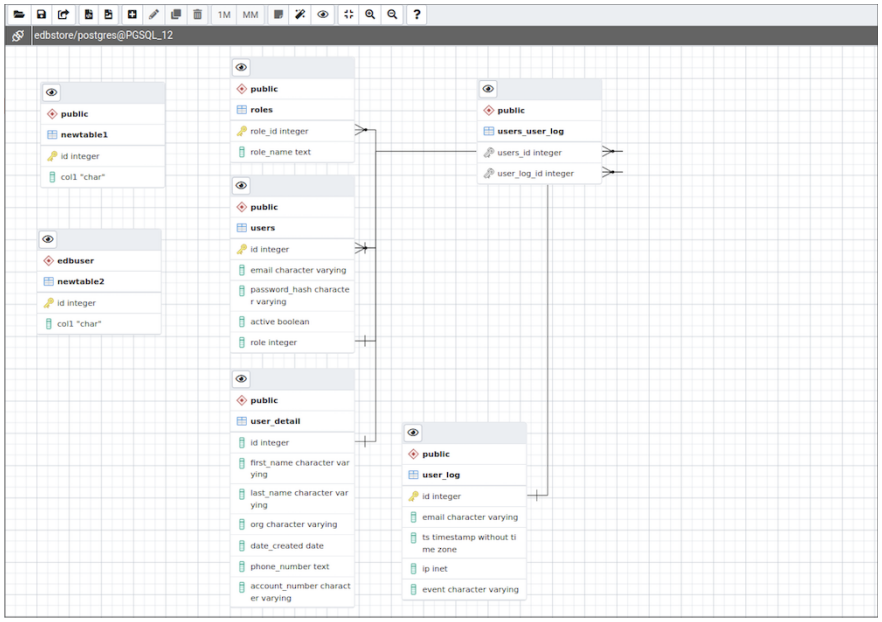


Fig. 1: ERD Tool

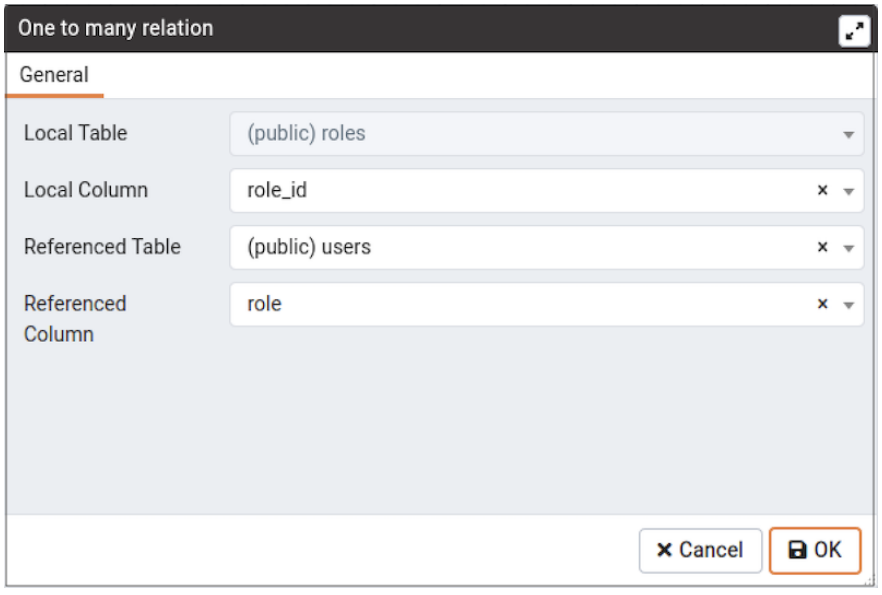


Fig. 2: ERD Tool - One-to-Many Dialog Box

- Select a Local Column that references another column.
- Select the Referenced Table. The Referenced Table has the `one` end point.
- Select the Referenced Column.

39.11 Many-to-many link dialog box

The many-to-many link dialog box allows you to:

- Add a many-to-many relationship between two tables.
- Create a relation table with linked columns that are derived from the two tables.
- Select the Left Table, which is the first table that is linked. The Left Table has the `one` endpoint of the link with the new relation table.
- Select the Left table Column, which is the first table's column. The Left table Column is a primary key.
- Select the Right Table, which is the second table that is linked. The Right Table has the `one` endpoint of the link with the new relation table.
- Select the Right table Column, which is the second table's column. The Right table Column is a primary key.

39.12 Table link

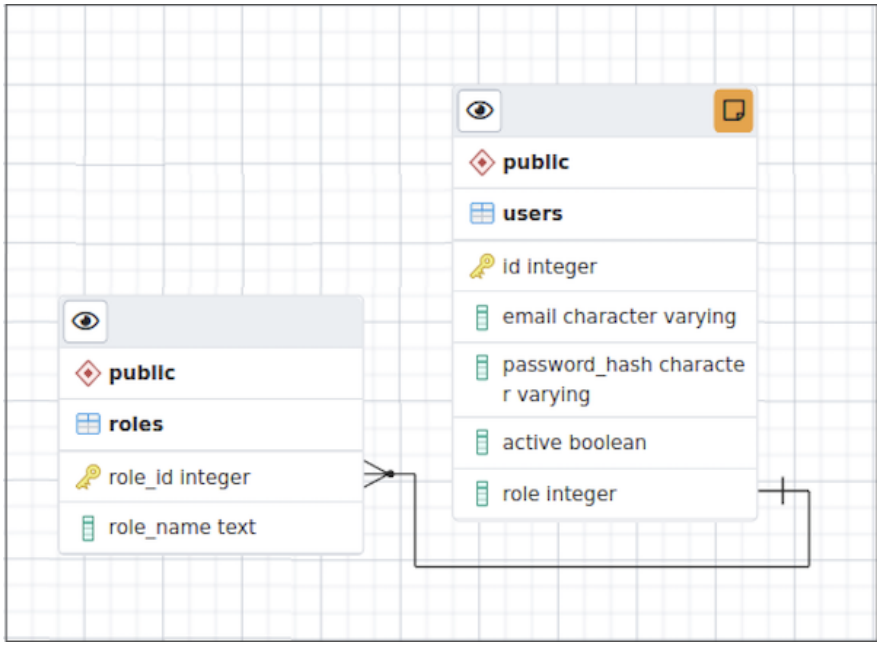


Fig. 3: ERD Tool - Table Link

The table link shows the relationship between tables:

- The single-line endpoint of the link shows the column that is the reference.
- The three-line endpoint of the link shows the column that references.
- If one of the columns that is a reference or that references is removed from the table, then the link drops.
- Select the link to drag it on the canvas.

39.13 Table notes

To add a note to a table:

1. On the toolbar, select **Add/Edit note**.
2. In the window, enter your note.

When a note is added to a table, **Note** becomes enabled on the Table node. To check or update notes, select **Note**.

CHAPTER 40

PL debugger

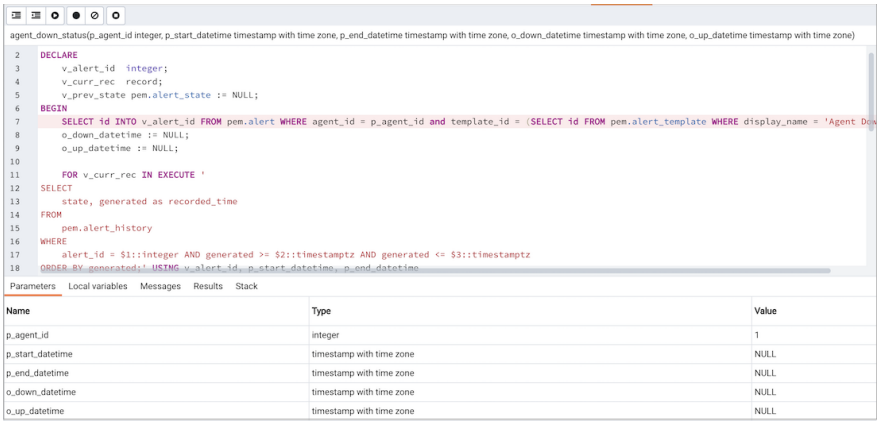


Fig. 1: Debugger page - Parameters tab

You can use the debugger to debug PL/pgSQL functions in PostgreSQL, EDB-SPL functions, stored procedures, and packages in EDB Postgres Advanced Server. The debugger is available as an extension for your PostgreSQL installation and is distributed as part of EDB Postgres Advanced Server. You need superuser privileges to use the debugger.

Before using the debugger, modify the `postgresql.conf` file, adding the server-side debugger components to the value of the `shared_preload_libraries` parameter:

```
shared_preload_libraries =\$libdir/`other_libraries`/plugin_debugger\
```

After modifying the `shared_preload_libraries` parameter, restart the server to apply the changes.

You can use the debugger for in-context debugging or direct debugging of a target function or procedure. When you use the debugger for in-context debugging, you set a breakpoint at the first line of a program. When a session invokes the target, control is transferred to the debugger. When using direct debugging, the debugger prompts you for any parameters required by the target and then allows you to step through the code.

40.1 In-context debugging

To set a breakpoint at the first line of a program, right-click the name of the object you want to debug, and select **Debugging > Set breakpoint**. The debugger window opens and waits for another session to invoke the program.

When another session invokes the target, the debugger displays the code, allowing you to add breakpoints or step through line by line. The other session is suspended until the debugging completes. Then control is returned to the session.

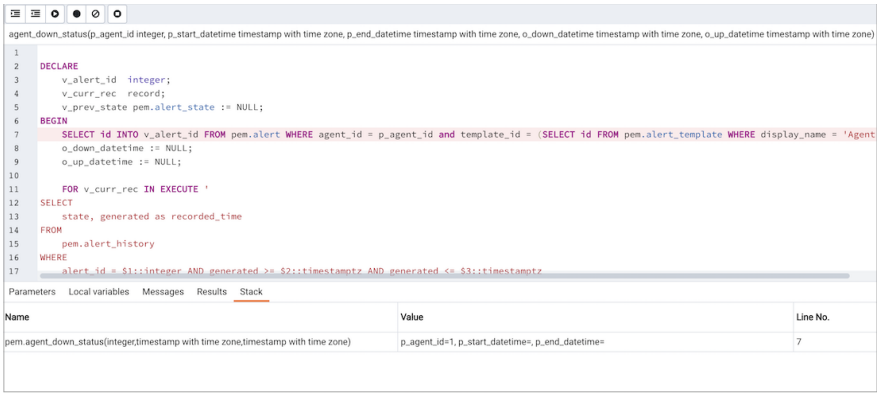


Fig. 2: Debugger page - Stack tab

40.2 Direct debugging

To use the debugger for direct debugging, in the browser tree control, right-click the name of the object that you want to debug and select **Debugging > Debug**.

Use the fields on the Debugger dialog box to provide a value for each parameter:

- The **Name** field displays the formal parameter name.
- The **Type** field contains the parameter data type.
- Select the **Null?** check box if the parameter is a NULL value.
- If the **Value** box contains an expression, select the **Expression?** check box.
- In the **Value** box, enter the parameter value to pass to the program. When entering parameter values, you can either:
- Type the value into the appropriate cell on the grid.
- Leave the cell empty to represent NULL.
- Enter two single quotes (") to represent an empty string.
- Enter a literal string consisting of just two single quotes (").
- PostgreSQL 8.4 and later supports variadic function parameters. You can enter these as a comma-delimited list of values, quoted or cast as required.
- Select the **Use default?** check box if you want the program to use the value in the **Default Value** box.
- The **Default Value** box displays the default value of the parameter.

After you enter the values the program requires, select **Debug** to start stepping through the program.

The values you selected are saved and appear the next time you open the dialog box. To clear the values, select **Clear All**.

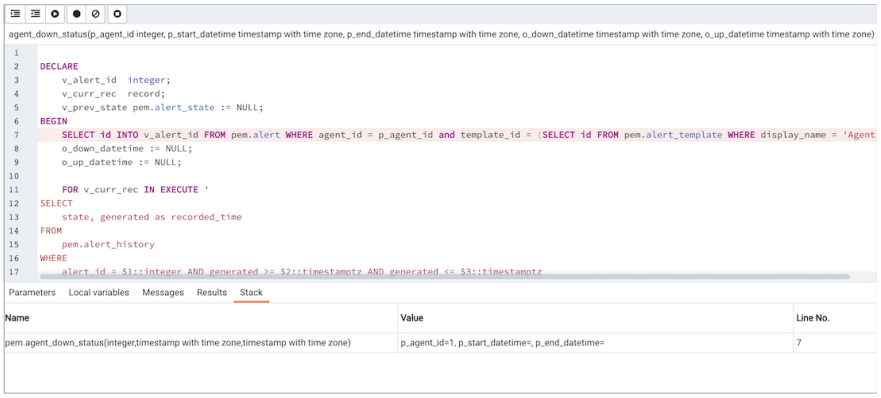


Fig. 3: Debugger page - Stack tab

40.3 Using the debugger

The main debugger window consists of two panels and a context-sensitive toolbar. Use toolbar icons to manage breakpoints and step into or through code.

Icon	Action
Step into	Execute the currently highlighted line of code.
Step over	Execute a line of code, stepping over any subfunctions invoked by the code. The subfunction executes but isn't debugged unless it contains a breakpoint.
Continue/Start	Execute the highlighted code and continue until the program encounters a breakpoint or completes.
Toggle breakpoint	Enable or disable a breakpoint without removing the breakpoint.
Clear all breakpoints	Remove all breakpoints from the program.
Stop	Stop the program from executing.

The top panel of the debugger window displays the program body. Click the gray margin next to a line number to add a breakpoint. The highlighted line in the top panel is the line that's about to execute.

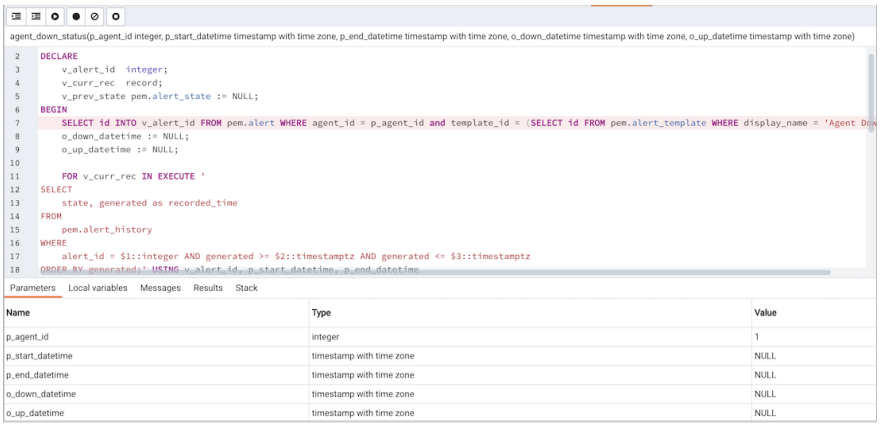


Fig. 4: Debugger main page

The lower panel of the debugger window provides a set of tabs that allow you to review information about the program:

- The **Parameters** tab displays the value of each parameter.
- The **Local** variables tab displays the current value of the program variables.
- The **Messages** tab displays any messages returned by the server.
- The **Results** tab displays the server message when the program completes.
- The **Stack** tab displays the list of functions that were invoked but that haven't yet completed.

As you step through a program, the **Local variables** tab displays the current value of each variable.

Fig. 6: Debugger page - Stack tab

Select a caller to change focus to that stack frame and display the state of the caller in the upper panel.

When the program completes, the **Results** tab displays the message returned by the server. If the program encounters an error, the **Messages** tab displays details.

!!!note If the `ENABLE_DEBUGGER` configuration option is set to false, then the debugger is disabled. !!!