



BeyondTrust

Privilege Management for Unix and Linux 22.2 Administration Guide

Table of Contents

Privilege Management for Unix and Linux Administration	9
Sample Policy Files	9
Privilege Management for Unix and Linux Features	10
Privilege Management for Unix and Linux Overview	12
Privilege Management for Unix and Linux Task Requests	13
Step-by-Step Task Processing	15
Normal Mode Processing	20
Optimized Run Mode Processing	21
Local Mode Processing	22
Privilege Management Virtualization	24
Privilege Management for Unix and Linux and AD Bridge	26
Privilege Management for Unix and Linux and BeyondInsight Console	28
Solr Indexing and Search	31
I/O Log Close Action	32
Splunk Integration	33
REST API for Privilege Management for Unix and Linux	36
Multi-Byte Character Set Support	37
PAM to RADIUS Authentication Module	38
Privilege Management for Unix and Linux Component, Directory, and File Locations ...	40
Sudo Wrapper	41
Settings	44
Base Directory	44
Client and Server Programs	44
Configuration and Settings Database	46
Role Based Policy	52
Role Based Policy Settings	60
Role Based Policy Entitlement Reports	63
Settings and Configuration Policy File Names	71
File Locations	72
Host and Port Specifications	77
Submit Task Requests to a Policy Server Daemon	81

Receive Task Requests from a Policy Server Daemon	85
Failover	88
Policy Server Host Connections to pblocald	93
Connections to SSH-Managed Devices	95
Connections to Log Servers	99
Host Name Verification	102
Control Connections	105
Port Usage	109
Auditing and Logging	114
Authorization Event Logging	115
Audit Events	123
Message Router	126
Change Management Events	131
License Events	132
File Integrity Monitor Events	133
Advanced Keystroke Action	134
Session Logging	137
I/O Logging	139
I/O Log Indexing and Searching	140
AD Bridge Event Logging	147
BeyondInsight Event Logging	150
BeyondInsight I/O Log Indexing and Searching	153
BeyondInsight Event and I/O Logging Common Settings	163
Event Queueing of Integrated Products	167
Log Synchronization	169
Diagnostic Logging	173
I/O Log Action	180
Log Archiving	183
Logfile Name Caching	187
Network Traffic and File Encryption	190
Set the Encryption Algorithm and Key	195
Pluggable Authentication Modules	210
Secure Socket Layers and Public Key Infrastructure	216

Kerberos Version 5	243
Privilege Management for Unix and Linux Shared Libraries	248
GUI Configuration	255
Other Security Issues	257
Licensing	264
REST Services	270
Scheduling Service	274
File Integrity Monitoring	275
Solaris Projects	277
Daemon Mode	281
Reporting	283
Manage and Test System Configuration	284
pbench	284
pbcheck	284
License Management	286
Client Limit Enforcement	288
Registry Name Service and Database Synchronization	292
Service Groups and Planning	292
Command Line Configuration	292
Settings and Configuration	295
Synchronize Policy Configuration and Other Configuration Files	305
Troubleshoot Registry Name Service and Database Synchronization Issues	308
Logging	312
Event Logs	313
Read Verbose Event Log Entries	314
Accept/Reject Logging	320
Keystroke Action Events	321
Reporting and Data Extraction	322
Merge Multiple Event Logs	323
Advanced Reporting	324
Event Log Rotation	325
I/O Logs	327
Diagnostic Logging	329

Debug Trace Logging	330
Log Synchronization	332
Log Cleanup and Rotation	333
Log Host File System Space	334
Log Archiving	335
Other Logging Limitations	337
Integrate Elasticsearch and Logstash	338
Configure Elasticsearch and Logstash	338
Integrate I/O Logging and Elasticsearch	352
Message Router Troubleshooting	355
Firewalls	357
TCP/IP Concepts and Terms	357
Privilege Management for Unix and Linux Connections	358
Privilege Management for Unix and Linux Shells	360
Processing	360
Shell Script Processing	362
Native Root Mode	363
Enable Debug Trace Logging for pbsh and pbksh	363
System Upgrades	364
Operating System and Hardware Upgrades	364
Network Upgrades	364
Administration Programs	366
pbbench	366
pbcall	369
pbcheck	371
pbdbutil, pbadmin	378
Role Based Policy Options	388
Client Registration Options	397
Management Event Options	399
REST Keystore Options	401
Registry Name Service Options	402
Database Synchronization Options	408
Registry Name Service Cache Options	409

File Integrity Monitor Options	410
Event Logfile Cache Options	412
I/O Logfile Cache Options	413
I/O Logfile Queue Options	415
Integrated Product Options	416
poldbg	417
Elasticsearch Credential Management	418
pbencode	420
pbguid	422
pbhostid	425
pbkey	426
pbless	428
pblicense	429
pblocald	430
pblog	433
pblogd	439
pbmasterd	441
pbmg	444
pbnvi	445
pbpasswd	446
pbping	448
pbprint	450
pbregister	452
pbreplay	454
pbreport	463
pbrun	465
pbssh	468
pbsum	471
pbsync	473
pbsyncd	476
pbumacs	478
pbuvqrpq	479
pbversion	480

pbvi	482
pblogarchive	483
Advanced Keystroke Action	486
Get Started with Advanced Keystroke Action	488
Maintenance and Configuration of Advanced Keystroke Action Policy	489
Advanced Keystroke Action Policy	490
Advanced Keystroke Action Logging / Events	501
File Integrity Monitoring	502
Overview	502
Policy Configuration	502
Predefinitions Section	502
Include Section	504
Exclude Section	504
Policy Configuration and Maintenance	505
Initial Setup and Configuration of the Service	506
REST API	510
Introduction	510
Architecture Overview	511
Functionality	512
Authentication	513
Installation and Configuration	514
Privilege Management for Unix and Linux REST API Programs	517
Specify a REST URL Path and Parameters	522
REST Calls and Parameters	523
Settings	524
License	526
Policy	530
IO Logs	536
Event Logs	541
File Integrity Monitoring (FIM)	546
Key File	549
Registry Name Service	551
Role Based Policy Database Manipulation	557

Role Based Policy - Miscellaneous Calls	558
Client Registration	562
SOLR	564
Elasticsearch Logstash API Calls	565
Java Implementation of REST API	567
X Window System Session Capturing	568
Introduction	568
Architecture Overview	569
X11 capture policy settings	569
Viewing Session Capture Output	570

Privilege Management for Unix and Linux Administration

This document shows Unix and Linux system administrators how to configure and use the Privilege Management for Unix and Linux software.

Privilege Management for Unix and Linux includes:

- Privilege Management for Unix and Linux
- Privilege Management for Networks
- Privilege Management for Unix and Linux, Essentials Edition



Note: This guide assumes that the user has a basic understanding of Unix or Linux system administration and some experience with a scripting or other computer language. We recommend that you have experience in these areas before you attempt to create or modify security policy files.



Note: Privilege Management for Unix and Linux refers to the product formerly known as PowerBroker for Unix and Linux.



Note: Specific font and line spacing conventions are used to ensure readability and to highlight important information, such as commands, syntax, and examples.

Sample Policy Files

When you install Privilege Management for Unix and Linux, you can choose to copy sample Privilege Management for Unix and Linux policy files to the installation host. These sample policy files include detailed explanations of what they do. You can use these files to learn how policy files are typically written for various scenarios. The directory that these sample files are copied to is determined by the GUI library directory option that you specify during installation. By default, this directory is `/usr/local/lib/pbuilder`. A `readme_samples` text file in that directory includes a brief description of each sample file.

Privilege Management for Unix and Linux Features

Privilege Management for Unix and Linux enables Unix and Linux system administrators to specify the circumstances under which users may run certain programs as root or as other important accounts. The result is responsibility for actions such as adding user accounts, fixing line printer queues, etc., allowing permissions to be safely assigned to the appropriate person without disclosing the root password. The full power of root is protected from potential misuse or abuse such as:

- Modifying databases
- File permissions
- Erasing disks

Furthermore, Privilege Management for Unix and Linux is capable of selectively recording all input and output from a session. Having this audit trail combined with the safe partitioning of root functionality provides an extremely secure means of sharing the root password. The **pbreplay** utility enables you to view sessions while they are happening or at a later date. Privilege Management for Unix and Linux can also require a checksum match before running any program, guarding against viruses or Trojan horse attacks on important accounts.

Through its integration with the SafeNet Luna SA Hardware Security Module (HSM), Privilege Management for Unix and Linux provides the first privileged user management solution to use FIPS 140-2 Security Level 2 encryption services to achieve compliance with the most strict key storage requirements and standards.

SafeNet Luna SA is a flexible, network-attached hardware security module that provides powerful cryptographic processing and hardware key management for applications where security and performance are a top priority. With FIPS 140-2 Security Level 2 and Common Criteria EAL 4+ validation, SafeNet Luna SA is an Ethernet-attached hardware security appliance that is designed to ensure the integrity and security of key management for PKI root key protection and smartcard issuance, with blazing cryptographic processing for digital signing applications or SSL acceleration for Web servers.

Privilege Management for Unix and Linux can access existing programs as well as its own set of utilities that execute common system administrative tasks. Utilities that are being developed to run on top of Privilege Management for Unix and Linux can manage passwords, accounts, backups, line printers, file ownership or removal, rebooting, logging people out, killing their programs, deciding who can log in to what machines from where, and so on. Users can work from within a restricted shell or editor to access certain programs or files as root.

Privilege Management for Unix and Linux can also optionally encrypt all network traffic that it generates, including control messages, input that is keyed by users, and output that is generated by commands that are run through Privilege Management for Unix and Linux. This encryption protects sensitive data from network monitoring.

Privilege Management for Unix and Linux is built upon two major concepts:

- Partitioning the functionality of root (and other important accounts) to allow multiple users to carry out system administration actions without full access to the administrative account or its password
- Creating an audit trail of such actions

Privilege Management for Unix and Linux enables system administration actions to be partitioned without compromising root account security. Privilege Management for Unix and Linux enables the system administrator to specify if, and when, a user's request to run a program is accepted or rejected.

Through Privilege Management for Unix and Linux, each user can request that a program be run on a machine as **root** (or as another important account, such as **oracle** or **admin**). Privilege Management for Unix and Linux evaluates the request. If the request is accepted, Privilege Management for Unix and Linux runs the program locally or across a network, on behalf of the user.

With Privilege Management for Unix and Linux, help desk personnel can reset passwords for users that have forgotten them and reinstate user accounts. Project members can clear a jammed line printer queue, kill hung programs, or reboot certain machines. Administration staff can print or delete resource usage logs or start backups.

Through partitioning, Privilege Management for Unix and Linux allows different users to perform those root actions for which they are responsible, but not anything else. Privilege Management for Unix and Linux enables you to specify:

- Which users can perform a particular task
- Which tasks can be run through the system
- When the user can perform the task
- On which machine the task can be performed
- From which machine the user may initiate a request to perform the task
- Whether another user's permission (in the form of a password) is required before the task is started
- The decisions that are to be made by a program that you supply and which Privilege Management for Unix and Linux calls to determine if a request should be accepted or rejected
- Many other miscellaneous properties of requests

Privilege Management for Unix and Linux is capable of recording all activity that passes through it to the I/O level. The power to accurately log activities in a safe environment enables you to implement a secure system administration regime with an audit trail. You always know exactly what is happening in root, as well as who did it, when it happened, and where.

Because root can modify any file, special precautions must be taken to ensure the Privilege Management for Unix and Linux logs are secure. Privilege Management for Unix and Linux can be configured to receive user requests from the submitting machine, execute tasks on the execution machine, and log all of the activities on yet another, very secure machine.

If necessary, the machines that contain the policy files and the log files can be made physically inaccessible to users and isolated from remote login over the network. In addition, the logs can be printed to hardcopy on a secure printer or recorded to a WORM drive if required.

This secure machine can also be assigned a root password, which is unknown to the person who has physical access to it, but known to someone else without physical access. Therefore, the two people would have to collude to subvert system security. These and other techniques may be used to achieve a high degree of security around Privilege Management for Unix and Linux, as well as the root activity logs that Privilege Management for Unix and Linux creates.

Privilege Management for Unix and Linux Overview

To effectively administer Privilege Management for Unix and Linux, it is necessary to understand how the product works. A typical Privilege Management for Unix and Linux configuration consists of the following primary components:

- **pbrun:** Used for secured task submission
- **pbmasterd:** Used for security policy file processing
- **pblocald:** Used for task execution
- **pblogd:** Used for writing event logs and I/O logs

It is possible to install any or all of these components on a single machine, or to distribute them among different machines. For optimal security, the policy server hosts and log hosts should be separate machines that are isolated from normal activity.

Privilege Management for Unix and Linux Task Requests

In the context of Privilege Management for Unix and Linux, there are two types of task requests:

- **Secured:** Requests must undergo security validation processing by Privilege Management for Unix and Linux before they can be run.
- **Unsecured:** Do not undergo security validation processing. These should be tasks that are not potential threats to the system and therefore do not fall under a company's security policy implementation. Unsecured tasks are handled by the operating system. Privilege Management for Unix and Linux is not involved in the processing of such tasks.

Secured Task Submission to SSH-Managed Devices - **pbssh**

Secured tasks can also be submitted through **pbssh**. **pbssh** is the Privilege Management component used to access SSH-managed devices where Privilege Management is not installed (routers, firewalls, Windows devices, or Unix/Linux devices where Privilege Management is not installed). **pbssh** connects to the target device using the SSH configuration.

Secured Task Submission - **pbrun**

All secured tasks must be submitted through **pbrun**, the Privilege Management for Unix and Linux component that receives task requests. A separate **pbrun** process is started for each secured task request that is submitted. If the use of **pbrun** is not enforced for secured tasks, then a company's security policy implementation could be compromised.



Note: *pbrun* must be installed on any machine from which a user can submit a secured task request.

Policy File Processing - **pbmasterd**

pbmasterd is responsible for applying the security rules (as defined in the Privilege Management for Unix and Linux policy files) which make up a company's network security policy. In other words, **pbmasterd** performs security verification processing to determine if a request is accepted or rejected based on the logic in the Privilege Management for Unix and Linux security policy files. If a request is rejected, then the result is logged and processing terminates. If a request is accepted, then it is immediately passed to **pblocald** for execution.

If **pblogd** is used, then **pbmasterd** terminates when the request is passed to **pblocald**. A separate **pbmasterd** process is started for each secured task request that is submitted. If the **pblogd** component is not being used, then **pbmasterd** waits for the **pblocald** process to complete before terminating.



Note: *During security verification processing, the first accept or reject condition that is met causes security policy file processing to immediately terminate. No further security verification processing is performed.*

If **pbmasterd** recognizes that a command is to be run on the host that submitted the request, then **pblocald** is optimized out of the connection. The command is run directly under the control of the client (that is, **pbrun**, **pbsh**, or **pbksh**), along with all logging and services that would have otherwise been provided by **pblocald**.

Task Execution - pblockd

pblockd executes task requests that have passed security verification processing (that is, requests that have been accepted by **pbmasterd**). After a task request is accepted, it is immediately passed from **pbmasterd** to **pblockd** in normal mode, or to **pbrun**, **pbsh**, or **pbksh** in local and optimized run modes. **pblockd** executes the task request as the user who is specified in the policy variable `runuser`, typically root or an administrative account. This action transfers all task input and output information back to **pbrun**.

In addition, **pblockd** logs pertinent task information to the Privilege Management for Unix and Linux event log (using **pbmasterd** or **pblogd**, depending on how Privilege Management for Unix and Linux has been deployed). The run host can also record task keystroke information to a Privilege Management for Unix and Linux I/O log (through **pbmasterd** or **pblogd**, depending on how Privilege Management for Unix and Linux has been deployed). A separate **pblockd** process is started for each secured task request that is submitted.

Logging - pblogd

pblogd is an optional Privilege Management for Unix and Linux component that is responsible for writing event and I/O log records. If **pblogd** is not installed, then **pbmasterd** writes log records directly to the appropriate log files rather than passing these records to **pblogd**. In addition, if **pblogd** is not installed, then **pbmasterd** must wait for the **pblockd** process to complete. If the **pblogd** is used, then **pbmasterd** terminates after task execution starts, and **pblockd** sends its log records directly to **pblogd**.

Using **pblogd** optimizes Privilege Management for Unix and Linux processing by centralizing the writing of log records in a single, dedicated component and eliminating the need for the **pbmasterd** process to wait for task execution to complete.

Logging - Message Router (pblighttpd-svc)

In Privilege Management for Unix and Linux v10.1.0, a new Message Router service was introduced to streamline the processing of events and other important messages throughout the system. It allows a single log server to quickly accept, process, and store tens of thousands of events every second.

Step-by-Step Task Processing

To make the following information more concise and easier to understand, this guide assumes that Privilege Management for Unix and Linux is installed on all of the machines that are involved. The guide also presumes that the network is functioning and there are sufficient resources (memory and disk space) to run the application and log what is required. Error processing within Privilege Management for Unix and Linux reports these problems when they occur.

This section describes the process that occurs when a task is submitted in Privilege Management for Unix and Linux, and indicates which modes use each part of the process.

There are three modes for Privilege Management for Unix and Linux:

- **Normal Mode:** In this mode, all tasks are performed including those that are run by **pblocald**.
- **Optimized Run Mode:** In this mode, after **pbmasterd** has accepted a request, the specified task runs directly on the submit host, without invoking **pblocald**. Doing this enables the administrator to use **pbmasterd** to validate a command, log the commands that are started in the event log, and record an I/O log for the secured task. The optimized run mode also reconfirms the password, performs time-out processing, and logs the status.
- **Local Mode:** In this mode, after **pbmasterd** has accepted a request, the specified task runs directly on the submit host, without invoking **pblocald**. This mode enables the administrator to use **pbmasterd** to authorize a command, log the accepted task. All other Privilege Management for Unix and Linux functionality is bypassed.

The following table summarizes the steps that are used for each of the three modes. An **X** represents a task that is processed by a specific mode, and **N/A** means that the task does not apply in the specified mode.

Process Task	Normal Mode	Optimized Run Mode	Local Mode
Secure task submitted	X	X	X
Policy Server daemon starts	X	X	X
Policy file processing	X	X	X
Local daemon started	X	N/A	N/A
Log daemon started	pblocald	pbrun	pbrun
pbrun/pblocald reconnect	X	N/A	N/A
runconfirmuser check	pblocald	pbrun	N/A
Executable check	pblocald	pbrun	pbrun
Secured task runs	X	X	X
Time-out processing	X	X	N/A
Secured task ends	X	X	X
pblocald completes	X	N/A	N/A
pblogd completes	Logs exit status and closes the I/O log	Logs exit status and closes the I/O log	Closes I/O log
pbmasterd completes	X	X	X
pbrun completes	X	X	X

Task Submitted (All Modes)

The initial step is for a user to execute **pbrun**. This is done either from the command line as:

```
pbrun list
```

or from a shell script such as:

```
#!/bin/sh
/usr/local/bin/pbrun list
```

where **list** is the task that is being requested. **pbrun** checks the settings file and sends the request with other information from the submit host to a policy server daemon that is specified in the **submitmasters** setting.

Policy Server Daemon Starts (All Modes)

The policy server daemon (**pbmasterd**) listens for requests from **pbrun**. When a request arrives, the policy server daemon checks its settings file. The policy server host settings file may be different from the settings file on the submit host because they may be on different machines. Validation that **pbrun** is trying to connect is performed and the rest of the policy server processing continues.

If there is an error at any point in the settings file validation or **pbrun** connect verification, then **pbmasterd** stops, and when possible, sends a message for the **pbrun** session to the user, and validates the client host name checks.

Policy File Processing (All Modes)

The main action of the policy server daemon is to confirm that the user may run a request, and to modify or set values for the request. Values can be set in the policy file that affect how the policy server daemon runs.

The values that are set in the policy file are shown in the following table:

Policy Values	Description
eventlog	Specifies the file in which the events are logged.
iolog	Identifies the file in which the I/O streams are logged.
localmode	Deprecated in favor of Optimized Run Mode Processing. This mechanism allowed execution on local host without the use of pblocald , with the expense of several features not available. Optimized Run Mode Processing enables all the features that localmode lacks, also without using pblocald .
lognopassword	Specifies whether passwords should be logged.
lognoreconnect	Identifies whether the log server should be allowed to run through pblocald or stay connected to pbmasterd and whether the pblocald should be allowed to connect to pbrun on submit host or stay connected to pbmasterd . In Optimized Run Mode, this has no affect.
noreconnect	Controls whether the policy server daemon should stay connected.

If necessary as part of the processing, the policy server daemon communicates with the **pbrun** session to get further information from the user, such as passwords or input.

If the log daemon is used and the **logmktemp ()** function is called, then **pbmasterd** starts the log daemon to create a log file on the log host. If the policy language variable **logignoreconnect** allows it, the log server reconnects to **pblocald** when the secured task is ready to run.

If the processing of the policy file reaches an accept statement, then **pbmasterd** tries to connect to **pblocald** on the run host.

If the processing of the policy file reaches a reject statement, then **pbmasterd** logs the result (possibly through the log server daemon) and terminates the request.

If the log daemon is being used, then **pbmasterd** tries to connect to the log daemon on the log host.

Privilege Management for Unix and Linux 8.0.2 adds a new **policytimeout** mechanism to protect against policies that appear nonresponsive.



Note: As soon as an **accept** or **reject** statement executes, policy file processing stops. No further policy file processing takes place.



For more information, please see the following:

- "Normal Mode Processing" on page 20
- "Optimized Run Mode Processing" on page 21
- "Local Mode Processing" on page 22
- "Optimized Run Mode Processing" on page 21
- "Timeout Processing (Normal Mode and Optimized Run Mode)" on page 19.

Local Daemon Started (Normal Mode)

The local daemon listens for requests from **pbmasterd**. When one arrives, it checks its settings file. The run host settings file may be different from the settings file on the policy server host because they can be on different machines. Validation that **pbmasterd** is trying to connect is performed and the rest of the local processing continues. The local daemon immediately determines whether it can accept requests from the policy server daemon by comparing the host to the **acceptmasters** line in the settings file.

If there is an error at any point in the settings file validation or the verification that **pbmasterd** is trying to connect the local daemon, then the process stops. When possible, a message is sent via the **pbmasterd** session for the **pbrun** session to the user. Validate policy server host name checks are also performed.

Log Daemon Started (All Modes)

The log daemon listens for requests from **pbmasterd** or **pblocald**. When one arrives, it checks its settings file. The log host settings file can be different from the settings file on the policy server host or run host because they can be on different machines. Validation that **pbmasterd** or **pblocald** is trying to connect is performed and the rest of the local processing continues.

If there is an error at any point in the settings file validation or the verification that **pbmasterd** or **pblocald** is trying to connect, then the log daemon stops. When possible, a message is sent using the requesting session for the **pbrun** session to the user. **pblocald** starts the log daemon in normal mode and **pbrun** starts the log daemon in local mode and optimization run mode.

pbrun/pblocald Reconnect (Normal Mode)

If **pbmasterd** does not need to stay in the middle of the connection between **pbrun** and **pblocald**, it instructs **pbrun** and **pblocald** to connect directly to each other. **pbmasterd** then exits.

pbmasterd removes itself when the following are all true:

- A log daemon is used.
- The **noreconnect** and **lognoreconnect** variables are **false**.

If these conditions are not met, then **pbmasterd** remains in the job stream and passes the data from **pbrun** to **pblocald**.

The only reason a policy server daemon would need to stay in the middle of a connection is that the policy server daemon is located between two subnets that do not normally allow traffic between them.

runconfirmuser Check (Normal Mode and Optimized Run Mode)

With all sessions now established, the **pblocald** session determines whether the **runconfirmuser** variable is set and requests the run host password for the **runconfirmuseruser** user from the **pbrun** session. If this request fails three times, then the **pblocald** session stops.

Executable Check (All Modes)

pblocald does some final checking before starting the actual command. If the **runcksum** or **runcksumlist** variable is set, **pblocald** determines whether the checksum of the command in **runcommand** matches the value in **runcksum** or **runcksumlist**. If the **runmd5sum** or **runmd5sumlist** variable is set, **pblocald** determines whether the MD5 checksum of the command in **runcommand** matches the value in **runmd5sum** or **runmd5sumlist**.

To log the checksum of the **runcommand** being compared against **runcksum**, **runcksumlist**, **runmd5sum**, or **runmd5sumlist**, use the policy variable **logcksum**.

These actions provide protection against viruses, Trojan horses, or other unintentional changes to the program file. **pblocald** also runs secure command checks. The final checking is done by **pblocald** for the normal mode and by **pbrun** in the optimized run mode and local mode.

Secured Task Runs (Normal Mode)

When the **pblocald** reaches this point, it can finally execute the command specified in the **runcommand** variable, **pblocald** checks that **runcommand** points to an executable file. If the file is not found or cannot be executed, **pblocald** stops and an error is sent back to the **pbrun** session.

pblocald sets up the run environment as follows:

- The runtime environment to execute the command is established according to the values in the **runenv** list.
- The user that is specified in the **runuser** variable runs the command.
- The **utmp** entry is written with the **runutmpuser** variable value as the user.
- The syslog is updated.
- The group is the value of the **rungroup** variable.
- The secondary groups are the value of the **rungroups** variable.
- The arguments to the command are the values that are specified in the **runargv** variable. The current directory is the value that is specified in the **runcwd** variable.

- The **umask** is the value of the **runumask** variable.
- The nice priority is the value of the **runnice** variable.
- If the **runchroot** variable is set, then the top of the file system is set via **chroot**
- The processing of HUP signals is set based on the value of the **runbkgd** variable.
- **pblocald** then starts the command.

Timeout Processing (Normal Mode and Optimized Run Mode)

If there is a **mastertimeout**, **submittimeout**, or **runtimeout** in effect (as specified in the policy or overridden by a client's **runtimeout** keyword in the settings), then the session terminates if there is no input or output activity within the specified number of seconds. These timeouts are effective only after the policy has accepted a request, during the lifetime of the secured task.

The Privilege Management for Unix and Linux 8.0.2 **policytimeout()** procedure provides a timeout mechanism that is effective during policy processing (before an accept or reject). This allows protection against **pbmasterd**/policy that appears nonresponsive waiting for user input, infinite loops within the policy, etc.

Secured Task Ends (All Modes)

At some point the task ends, because the command finished, the user interrupted it by pressing **CTRL+C**, or it was exited in some other way.

pblocald Completes (Normal Mode)

pblocald recognizes task completion and stops processing. It captures the reason for the completion (such as a signal or an exit code) and sends it for logging as the **exitstatus** variable. The **exittime** and **exitdate** are also logged. In normal mode, **pblocald** completes.

pblogd Completes (All Modes)

If a log server is used, then the I/O log is closed. For normal mode and optimized run mode, the exit status of the secured task is also logged.

pbmasterd Completes (Normal Mode Only)

If the **pbmasterd** session is still running, then it shuts down. The **pblogd** session also shuts down.

pbrun Completes (Normal Mode and Optimized Run Mode)

pbrun displays the **exitstatus** of the string of the secured task if the task detects an error or abnormal exit.

The exit status of the secured task is also returned in the **pbrun** exit status value.

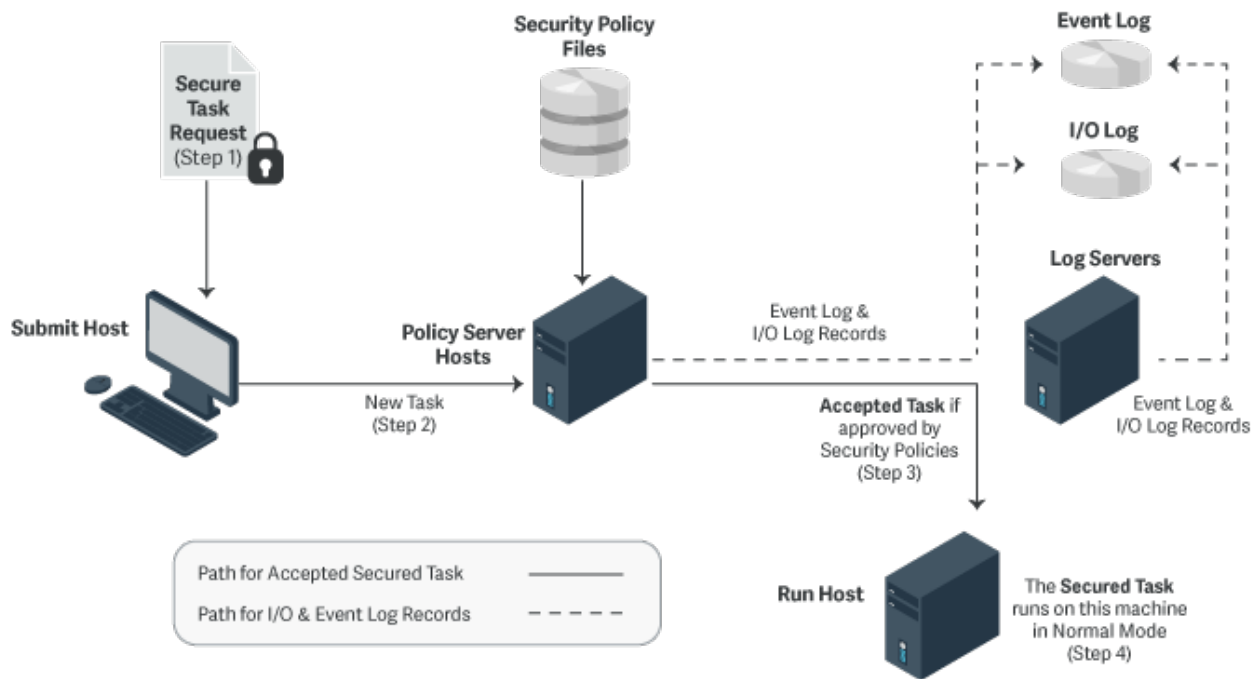


For more information, please see your Unix/Linux man pages.

Normal Mode Processing

As shown in the following figure, in normal mode, the machine from which a task is submitted is referred to as the *submit host*. The machine on which security policy file processing takes place is referred to as the *policy server host*. The machine on which a task is actually executed is referred to as the *run host*. The machine on which event log records and I/O log records are written is referred to as the *log host*. Use of the log server daemon **pblogd** is optional, but highly recommended.

How Privilege Management for Unix and Linux Works

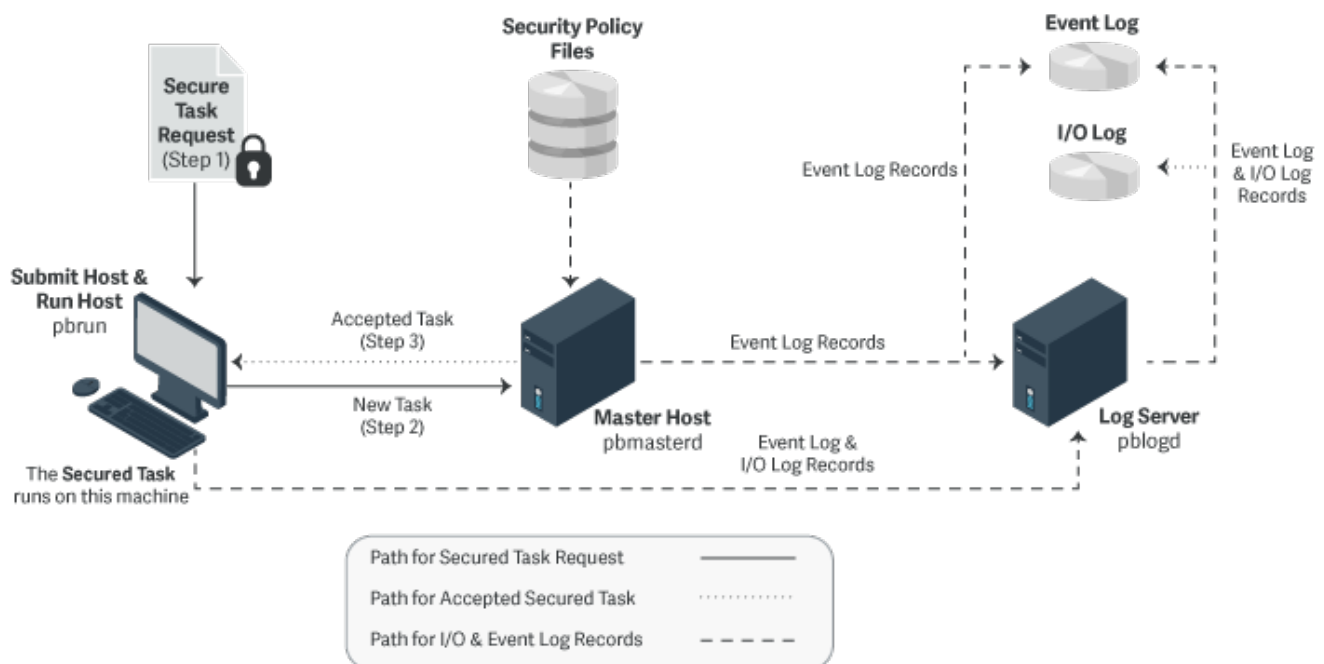


Optimized Run Mode Processing

- **Version 3.5 and earlier:** Optimized run mode not available.
- **Version 4.0 and later:** Optimized run mode available.

In optimized run mode, after **pbmasterd** has accepted a request, the specified task runs directly on the submit host, without invoking **pblocald**. This feature enables the administrator to use **pbmasterd** to validate a command, log the commands that were started in the event log, and log the I/O streams for the secured task. The optimized run mode also reconfirms the password, performs time-out processing, and logs the status.

The following figure illustrates the processing when Privilege Management for Unix and Linux is running in the optimized run mode:



Optimized Run Mode Availability

Optimized run mode is enabled when all of the following conditions are met:

- The policy server host is configured to use a log server.
- The values of the **submithost** and **runhost** variables must be equal.
- **pbrun** is invoked without the **--disable_optimized_runmode** command line option.
- **pbmasterd** is invoked without the **--disable_optimized_runmode** command line option.
- The settings file on the submit host has the **clientdisableoptimizedrunmode** setting set to **no**.
- The settings file on the policy server host has the **masterdisableoptimizedrunmode** setting set to **no**.
- The policy sets the **runoptimizedrunmode** variable to **true**.

Local Mode Processing

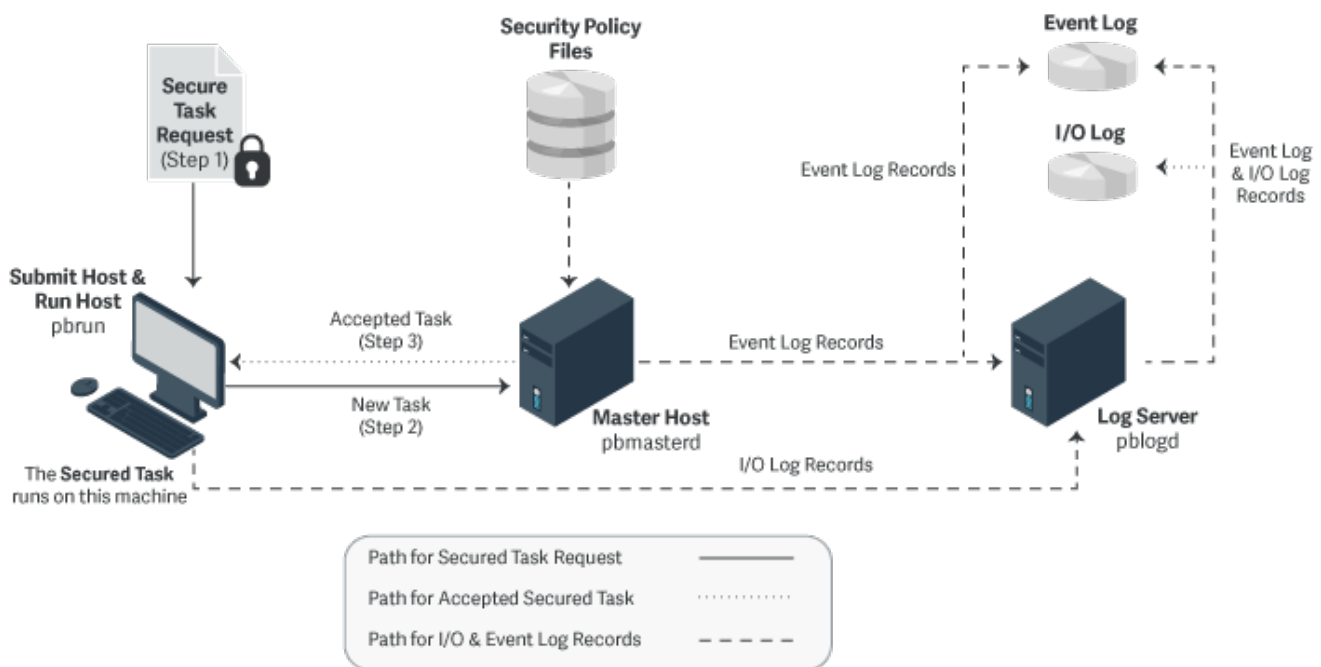
Deprecated in favor of Optimized Run Mode.

In local mode, after **pbmasterd** has accepted a request, the specified task runs directly on the submit host, without invoking **pblocald** and without using Optimized Run Mode. This feature enables the administrator to use **pbmasterd** to authorize a command and to log the accepted task in the event log. However, unlike optimized run mode, this mode does not perform timeout processing, log the exit status of the accepted task, or Advanced Control and Audit (ACA). In local mode, the process **pbrun** is replaced by the secured task, unless the I/O Logging is on.

With the introduction of optimized run mode, the use of local mode is no longer a benefit, since optimized run mode allows the task to run without invoking **pblocald** (if on the same host), and allows time-out processing, I/O logging, logs the exit status of the task, and allows ACA.

Local mode processing can be controlled in the `/etc/pb.settings` file (**allowlocalmode** setting) or in the policy (**localmode** and **runlocalmode** variables).

The following figure illustrates the processing when Privilege Management for Unix and Linux is running in local mode.



For more information on Optimized Run Mode, please see "Optimized Run Mode Processing" on page 21.

Local Mode Availability

Local mode is enabled when the **allowlocalmode** setting on the submit host, policy server host, and run host is set to **yes**.



Note: *Deprecated in favor of Optimized Run Mode. For more information on Optimized Run Mode, please see "Optimized Run Mode Processing" on page 21. **pbrun** must be invoked with the **-l** command line option, or the policy must set the **runlocalmode** variable to true.*

Local Mode Effects

Local mode does the same processing on the submit host and the policy server host, including the logging of the accepted request. However, instead of the policy server daemon requesting the **pblocald** to run the task, the **pbrun** session is replaced with the task. **pblocald** is not run when using local mode. **pblogd** may be run to record the *Accept* event.

In local mode, the accepted task runs on the submithost. Local mode fails with an error if a different runhost is specified.

Local Mode Limitations

Because the Privilege Management for Unix and Linux programs are not active when a program runs with local mode, the following limitations exist:

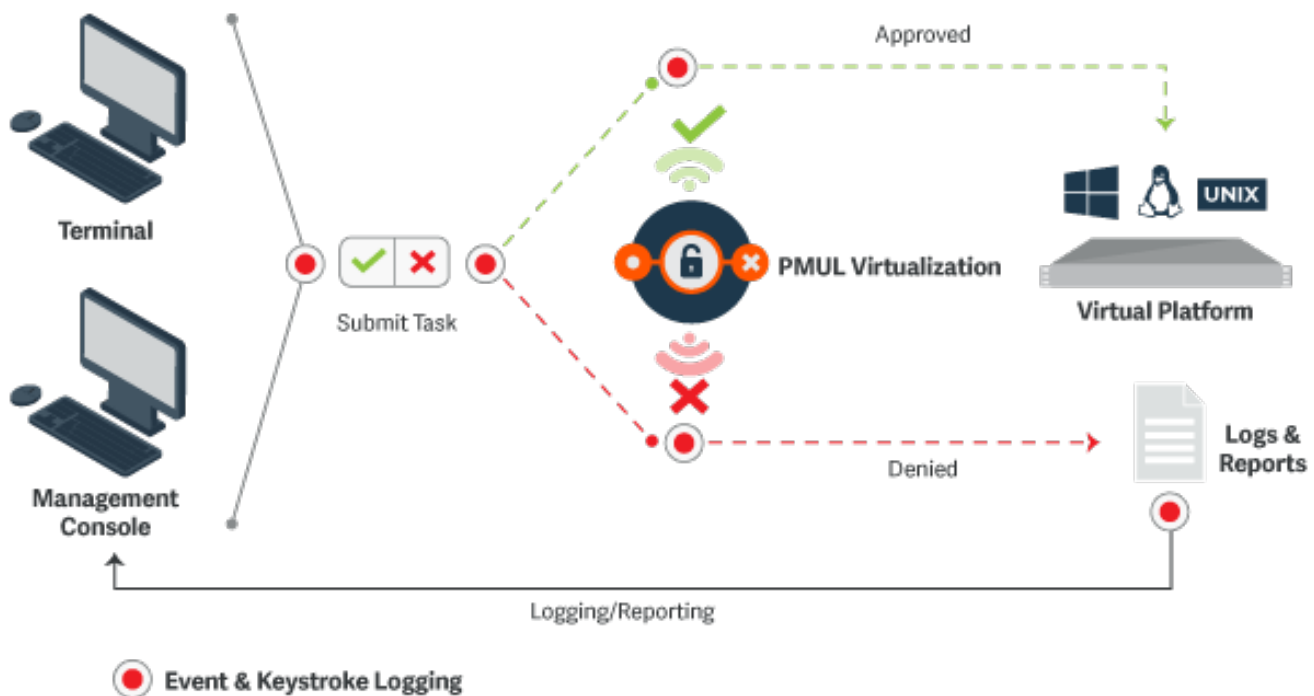
- Exit status of the job is not logged.
- **runtimeout** and **submittimeout** cannot be processed.
- Keystroke actions cannot be processed.
- The program specified by **iologcloseaction()** policy procedure is not executed.
- ACA is not compatible with local mode.

Privilege Management Virtualization

Privilege Management Virtualization is a cost-effective solution for consistent granular privilege identity management across guest operating systems as well as hypervisor hosts. Privilege Management Virtualization provides granular delegation of administrative privileges on virtual guest and host hypervisors, including detailed and flexible reporting with keystroke logging of administrative actions, for a secure and compliant virtualized datacenter environment.

Privilege Management Virtualization enables organizations that move to virtualized platforms to control administrative access to the Hypervisor/VMM layer while still realizing all virtualization cost efficiencies. Administrative tools prevent the virtualization layer from being compromised, possibly posing significant security risks to all hosted workloads. Programmable role-constraint mechanisms enforce segregation of duties for users and virtual platform-specific, cost-effective deployment capabilities enable secure datacenter virtualization.

The following diagram shows how Privilege Management Virtualization works.



Privilege Management Virtualization Features

The features of Privilege Management Virtualization include:

- Automated workflows for policy creation and change management
- Granular delegation of administrative privileges
- Detailed and flexible reporting including keystroke logging of administrative activities
- Two-click entitlement reports
- Programmable role-constrain mechanisms for segregation of duties

- Secures virtual guest and host hypervisors
- Supports VMware ESX, Solaris Zones, AIX WPAR, and IBM z/VM

Privilege Management for Unix and Linux and AD Bridge

Starting with v7.0, Privilege Management for Unix and Linux can be integrated with AD Bridge. Integrating Privilege Management for Unix and Linux with AD Bridge has the following benefits:

Event Log Central Collection

AD Bridge features a database-centric reporting architecture that enables event collection from multiple devices as well as the ability to report about this data from a central location using standard plug-ins. Events are forwarded to the AD Bridge collector machines where AD Bridge tools are installed and are running **BTCollector** services. The collector machines then aggregate all events in an enterprise-wide MS SQL Server database.

Starting with v7.0, Privilege Management for Unix and Linux incorporates the collection of Privilege Management for Unix and Linux events (Accept events, Reject events, Finish events, and Keystroke Action events) by AD Bridge collectors as well as the ability to query this information using the standard AD Bridge report plug-in.

The AD Bridge Enterprise tools on Windows include a management console, which supports a number of plug-ins for performing various tasks. The reports plug-in has many available reports for viewing configuration and event related queries. Privilege Management for Unix and Linux now has dedicated reports for the various operations that it performs.

Privilege Management for Unix and Linux Health Check

Starting with v7.0, Privilege Management for Unix and Linux, as part of integration with AD Bridge, sends events to AD Bridge Collectors based on the responsiveness of Privilege Management for Unix and Linux policy server hosts, log hosts, and **pblocald**. Privilege Management for Unix and Linux clients **pbrun**, **pbsh**, **pbksh**, and **pbssh**, optionally report a new failover event every time a Privilege Management for Unix and Linux policy server host or log host fails to respond in a timely manner.

This feature is closely tied to the current Privilege Management for Unix and Linux failover mechanism. Any policy server that does not respond within the number of seconds specified by the **masterdelay** setting causes the new failover event to be written to both syslog and the AD Bridge event log database. Any log host that does not respond within the number of seconds specified by the **logserverdelay** setting causes the new failover event to be written to both syslog and the AD Bridge event log database. Similarly, **pbmasterd** reports events any time **pblocald** fails to respond. This new feature also allows for the optional recording of successful connection events.

Another plug-in for the management console is the Privilege Management Operations Dashboard. This is a tool that provides a view on key metrics that an administrator can configure to show green, yellow, and red status indicators depending on user-defined thresholds. The health events are illustrated on this Privilege Management Operation Dashboard using the colors green, yellow, and red to indicate status.

Integration Process

To enable AD Bridge to work with Privilege Management for Unix and Linux, the AD Bridge agent must be installed on the appropriate Privilege Management for Unix and Linux machines:

- On the Privilege Management for Unix and Linux policy server host and log host computers to send the event log records (Accept, Reject, Finish, and Keystroke Actions events) and the health event log records (related to **pblocald**) to AD Bridge.
- On the client computers (where **pbrun**, **pbksh** or **pbsh**, and **pbssh** are installed), policy server host, and run host (where **pblocald** is installed), to send the health event log records related to the policy server host and log host to AD Bridge.

To send Privilege Management for Unix and Linux event logs to AD Bridge, you must set the following in the **pb.settings** file:

- **sharedlibpbisdependencies**
- **pbis_event_logging**

To send event records about the health of the policy server host, log host, and **pblocald** to the AD Bridge, you must set the following in the **pb.settings** file:

- **sharedlibpbisdependencies**
- **pbis_log_failover**
- **pbis_log_connect_success**



For more information, please see the following:

- *For instructions on how to install the products:*
 - *[AD Bridge Installation Guide](https://www.beyondtrust.com/docs/ad-bridge/getting-started/index.htm) at <https://www.beyondtrust.com/docs/ad-bridge/getting-started/index.htm>*
 - *[Privilege Management for Unix and Linux Installation Guide](https://www.beyondtrust.com/docs/privilege-management/unix-linux/index.htm) at <https://www.beyondtrust.com/docs/privilege-management/unix-linux/index.htm>*
 - *the Privilege Management for Unix and Linux readme file*
 - *the AD Bridge readme file for more information about supported platforms*
- *"sharedlibpbisdependencies" on page 149*
- *"AD Bridge Event Logging" on page 147*
- *"pbis_log_failover" on page 148*
- *"pbis_log_connect_success" on page 148*

Privilege Management for Unix and Linux and BeyondInsight Console

Starting with v7.5, Privilege Management for Unix and Linux can be integrated with BeyondInsight. This integration with BeyondInsight has the following benefits:

Event Log Central Collection

BeyondInsight features a database-centric reporting architecture that enables event collection from multiple devices as well as the ability to report about this data from a central location.

Privilege Management for Unix and Linux uses a message router architecture to store its events (Accept events, Reject events, Keystroke Action events, and Finish events) in a database, defined by **integratedproductsqueuedb**, and uses a scheduler to periodically forward these event log records to BeyondInsight Web services. Using BeyondInsight, you can then sort and filter this data into useful reports. BeyondInsight also uses these events to display the list of Privilege Management for Unix and Linux servers in the BeyondInsight Assets.

I/O log Indexing for Improved Search Capabilities

This integration allows BeyondInsight to search for I/O logs via an indexed search. Privilege Management for Unix and Linux uses Solr (with Lucene and Jetty) to index I/O log output data and BeyondInsight performs Solr queries and interprets/displays the results, allowing the user to replay the resulting I/O logs via **pbguid**.

Integration Process

When installing or upgrading, you can enable BeyondInsight Integration and provide values to the necessary keywords to enable these features.

To enable Privilege Management for Unix and Linux v7.5 (and later) to send the event log records to BeyondInsight, perform the following actions.

1. Copy and convert the BeyondInsight client certificate from BeyondInsight to the policy server and log server hosts.
 - Start the BeyondInsight Configuration Tool on the BeyondInsight management console server. Click **Generate Certificate Zip** in the BeyondInsight Configuration Tool.
 - Select the output folder for the ZIP file and a password to apply to the exported **.pfx** file.
 - Select a folder where you can securely copy the **certificates.zip** file and uncompress it.
 - Copy the following files to a secure directory on the Privilege Management policy server and log server hosts:
 - **eEyeEMSClient.pem**
 - **<host>_eEye_EMS_CA.pem**
 - In the Privilege Management servers and log server settings file, assign the keyword **sslrcscertfile** to the location of **eEyeEMSClient.pem**:

```
sslrcscertfile /<secure_directory>/eEyeEMSClient.pem
```

- Assign the keyword **sslrcscafile** to the location of the file **<host>_eEye_EMS_CA.pem**:

```
sslrcscafile /<secure_directory>/<host>_eEye_EMS_CA.pem
```

- Start the BeyondInsight Configuration Tool on the BeyondInsight management console server. Click **Generate Certificate Zip** in the BeyondInsight Configuration Tool.
- Select the output folder for the ZIP file and a password to apply to the exported **.pfx** file.
- Select a folder where you can securely copy the **certificates.zip** file and uncompress it.
- Copy the following files to a secure directory on the Privilege Management policy server and log server hosts:
 - **eEyeEMSClient.pem**
 - **<host>_eEye_EMS_CA.pem**
- In the Privilege Management servers and log server settings file, assign the keyword **sslrcscertfile** to the location of **eEyeEMSClient.pem**:

```
sslrcscertfile /<secure_directory>/eEyeEMSClient.pem
```

- Assign the keyword **sslrcscafile** to the location of the file **<host>_eEye_EMS_CA.pem**:

```
sslrcscafile /<secure_directory>/<host>_eEye_EMS_CA.pem
```



Note: In the Privilege Management settings file, assign the keyword **rcshost** to the BeyondInsight hostname exactly how it appears in this certificate.

2. If you have not done so during the installation, set the following keywords in **pb.settings** on the policy server and log server hosts:
 - **rcshost**
 - **rscwebsvcport**
 - **sslrcscertfile**
 - **sslrcscafile**

To enable Privilege Management for Unix and Linux v7.5 and above to index I/O log files:

1. Install Solr on a server of your choice.
2. Copy SSL certs from the Solr server to policy server and log server hosts.
 - On the Solr server, securely copy the PEM encoded files (***.pem**) from the **etc** directory under Solr install directory (for example **/opt/pbul-Solr/etc**) directory to the directories pointed by **Solrcafile**, **Solrclientkeyfile**, and **Solrclientcertfile** on the desired log server and policy server hosts.
3. If you have not done so during the installation, set the following keywords in **pb.settings** on the policy server and log server hosts:
 - **pbreplaylog**
 - **Solrhost**
 - **Solrport**
 - **Solrcafile**
 - **Solrclientkeyfile**
 - **Solrclientcertfile**
 - **integratedproductsqueuedb**

4. Other keywords to set (common to both event log collection and I/O log indexing):

- **pbadminpath**
- **guiport**
- **sguiport**
- **sharedlibkrb5dependencies**
- **sharedlibssldependencies**
- **sharedlibldapdependencies**
- **sharedlibcurldependencies**



For more information, please see the following:

- [BeyondInsight documentation](https://www.beyondtrust.com/docs/beyondinsight-password-safe/index.htm) at <https://www.beyondtrust.com/docs/beyondinsight-password-safe/index.htm>
- For more information about the **pb.settings** keywords:
 - "BeyondInsight Event Logging" on page 150
 - "BeyondInsight I/O Log Indexing and Searching" on page 153
 - "BeyondInsight Event and I/O Logging Common Settings" on page 163
- For installing Solr, Solr Installation in the [Privilege Management for Unix and Linux Installation Guide](https://www.beyondtrust.com/docs/privilege-management/unix-linux/index.htm) at <https://www.beyondtrust.com/docs/privilege-management/unix-linux/index.htm>
- For the names of the PEM encoded files to copy when enabling Privilege Management for Unix and Linux to index IO log files, the Post-install section of Solr Installation in the [Privilege Management for Unix and Linux Installation Guide](https://www.beyondtrust.com/docs/privilege-management/unix-linux/index.htm) at <https://www.beyondtrust.com/docs/privilege-management/unix-linux/index.htm>.

Solr Indexing and Search

There are separate tar files for Solr installation. Each log server and policy server host is able to communicate with a Solr server and submit I/O log output data for indexing. BeyondInsight and BIUL (BeyondInsight for Unix & Linux) provide a search GUI, allowing users to search indexed I/O logs using a selected set of variables, but also allowing to search the content of I/O log sessions using queries such as *this AND that AND NOT other OR somethingelse*.

For each I/O log file, the result of **pbreplay -O** output of the I/O log file is sent to Solr to be indexed. Some of the event log variables in the header of the I/O log are indexed as well. These variables are:

- **user**
- **runuser**
- **runhost**
- **runcommand**
- **runargv**

The name of the I/O log file name is also indexed, as well as the start and end time of the I/O log session.

You can add user-defined eventlog variables (defined in the policy) to the list of variables to be indexed by setting **Solrvariables** in **pb.settings** to the list of user variables defined in the policy. These variables must be named **<var>_pbul**.

The result displayed contains a path to the actual I/O log file, which can then be replayed using Privilege Management for Unix and Linux GUI (this requires Privilege Management GUI to be installed on the log server and policy server hosts where the I/O log files reside).

If I/O log indexing with Solr is enabled, the Solr index is updated when I/O logs get archived.

If a problem occurs while trying to contact the Solr server (broken connection, miscellaneous errors, etc.), an appropriate error is logged in the diagnostics log file, and the unsent I/O log file name is saved to be forwarded to the Solr server at a later time.

Privilege Management periodically checks to see if there are events that are outstanding, and are older than the **autofwdtime** setting. If conditions are met, it launches the **pbreplay** admin binary to forward the I/O log data to the Solr server for indexing. The path where **pbreplay** resides is specified by the setting keyword **pbadminpath**.

Starting with Privilege Management for Unix and Linux v10.0.0, a queue mechanism is used to process I/O logs for Solr, while limiting the number of indexing processes. This mechanism is shared by the feature I/O Log Close action.



For more information, please see the following:

- ["Solrvariables" on page 154](#)
- ["I/O Log Close Action" on page 32](#)

I/O Log Close Action

The new Privilege Management policy procedure **iolocloseaction** allows the policy to specify a program that is executed for each completed iolog.

This mechanism allows the I/O log to be processed in some way determined by the specified program. For example, Privilege Management includes a perl script that sends ACA data from the I/O log to Splunk.

The **iolocloseaction** mechanism and the Solr indexing mechanism share a queue that allows **pbconfigd** to control and monitor **pbreplay** processes, which in turn perform the Solr indexing and **iolocloseaction** actions. This mechanism uses a combination of fast write to queue files, and a database. Each I/O logging process writes the **iolog** path and filename to the queue, as well as periodic heartbeat information to inform the queue mechanism that the I/O log is still being generated. When an **iolog** is closed (normally), that information is written to the queue as well. **pbconfigd** runs a scheduled task that transfers data from the IO Log Action queue to both **SOLR** and any specified IO Log Close Action scripts and once they have been successfully processed the entry is deleted. The scheduler automatically tries to resend any outstanding entries if the **SOLR** service is down or unavailable.

Both Solr indexing and **iolocloseaction** are ultimately processed by **pbreplay**. **pbconfigd** runs a scheduled task that monitors the **pbreplay** processes handling Solr indexing and/or **iolocloseaction**. The number of allowed **pbreplay** processes is configured with the **iologactionmaxprocs** keyword. **pbreplay** processes are launched as needed to process the database queue. The **iologactionretry** keyword controls the number of retries to acquire a database lock for the internal database queue operations.



For more information, please see the [Privilege Management for Unix and Linux Policy Language Guide](https://www.beyondtrust.com/docs/privilege-management/unix-linux/index.htm) at <https://www.beyondtrust.com/docs/privilege-management/unix-linux/index.htm>.

Splunk Integration

Privilege Management can send Accept and Reject event data to Splunk via syslog, using the `syslogsession_start_format` and `syslog_reject_format` settings.

Privilege Management for Unix and Linux 10.0.1 adds a new keyword `syslogsession_finished_format_logserver`, which adds exit status data, and operates from the log server (as opposed to the `syslogsession_finished_format` keyword that operates from each runhost). Both syslog and syslogsessions must be set to **yes** to enable those keywords. The syslog keyword needs to be configured to send data to Splunk.



Note: Various syslog implementations have data rate limiting and must be configured accordingly.

Privilege Management can also send ACA data to Splunk, via the `iologcloseaction()` procedure defined in the policy language. This makes use of the Perl script `closeactionsplunk.pl`, normally located in `/opt/pbul/scripts/`.



Note: The use of this Perl script may require additional Perl modules to be installed. This script requires a Privilege Management for Unix and Linux REST App ID and App Key to be configured near the top of the script.

Example Splunk App

Privilege Management for Unix and Linux has an example Splunk app available from the Splunk website.

Once the Splunk App is installed in Splunk, if Splunk is to be configured to accept syslog data, do the following within the Splunk GUI:

1. Click **Settings > Data Inputs > UDP + Add New**.
2. Enter port **514**, then click **Next**.
3. Click **App Context**.
4. Select **BeyondTrust App for Splunk (App-BeyondTrust)**.
5. Click **Select Source Type**.
6. Enter the first few characters of **beyondtrust:syslog**. The search box should find **beyondtrust:syslog**.
7. Select that, click **Review**, and then **Submit**.
8. Click **Settings > Advanced Search > Search Macros**.
9. Select the app: **BeyondTrust App for Splunk**.
10. Verify that the macro named `get_beyondtrust_index_sourcetype` has the **Definition: (index="main" sourcetype="beyondtrust:syslog")**.

To send Reject and Finish event data to Splunk (in a format that the Splunk app recognizes), set the following syslog formatting keywords in `/etc/pb.settings` on the policy servers and log servers:

```
syslog_reject_format "BeyondTrust_PBUL_REJECT_Event: Time_Zone='%timezone%'; Request_Date='%date%'; Request_Time='%time%'; Request_End_Date='%date%'; Request_End_Time='%time%'; Submit_User='%user%'; Submit_Host='%submithost%'; Submit_Host_IP='%submithostip%'; Run_User='None'; Run_Host='None'; Run_Host_IP='No IP Address'; Current_Working_Directory='%cwd%'; Requested_Command='%command%'; Requested_Arguments='%argv%'; Command_Executed='None'; Command_Arguments='%runargv%'; ACA_Event='False'; ACA_Date='NA'; ACA_Time='NA'; ACA_Authorization='NA'; ACA_CWD='NA'; ACA_Action='NA'; ACA_Target='NA'; ACA_Arguments='NA'; Log_Servers='%logservers%'; Session_Recording_File='Session Not Recorded'; Risk_Rating='%pbrisklevel%'; Authorizing_
```

```
Server='%masterhost%'; Event_Status='Reject'; Exit_Status='%exitstatus%'; Risk_
Rating='%pbrisklevel%'; Authorizing_Server='%masterhost%'; Event_Status='Reject'; Exit_
Status='%exitstatus%'"
```

```
syslogsession_finished_format_logserver "BeyondTrust_PBUL_ACCEPT_Event: Time_Zone='%timezone%';
Request_Date='%date%'; Request_Time='%time%'; Request_End_Date='%exitdate%'; Request_End_
Time='%exittime%'; Submit_User='%user%'; Submit_Host='%submithost%'; Submit_Host_
IP='%submithostip%'; Run_User='%runuser%'; Run_Host='%runhost%'; Run_Host_IP='%runhostip%';
Current_Working_Directory='%cwd%'; Requested_Command='%command%'; Requested_Arguments='%argv%';
Command_Executed='%runcommand%'; Command_Arguments='%runargv%'; ACA_Event='False'; ACA_Date='NA';
ACA_Time='NA'; ACA_Authorization='NA'; ACA_CWD='NA'; ACA_Action='NA'; ACA_Target='NA'; ACA_
Arguments='NA'; Log_Servers='%pblogdnodename%'; Session_Recording_File='%iolog_list%'; Risk_
Rating='%pbrisklevel%'; Authorizing_Server='%masterhost%'; Event_Status='Accept'; Exit_
Status='%exitstatus%'"
```

The log servers require the **-r** option to syslog rejects.

For example, on RHEL 6.x, edit **/etc/xinetd.d/pblogd**, changing **server_args** to include the **-r**, then restart **xinetd**.



Example:

```
server_args = -r -i
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

Similarly, for RHEL 7.x, edit **/etc/systemd/system/pblogd@.service** so that **ExecStart** includes the **-r**, and restart the **pblogd** service.



Example:

```
ExecStart=-/usr/sbin/pblogd -r -i
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
```

To send ACA data to Splunk (in a format that the Splunk app recognizes), the policy must specify an I/O log and enable session history as well as specify an **iologcloseaction** to run the Perl script. The example Privilege Management for Unix and Linux policy **/opt/pbul/policies/pbul_functions.conf** includes example **Procedure SplunkRole()** to accomplish all the necessary tasks (to enable this procedure, set **EnableSplunkRole = true** in **/opt/pbul/policies/pbul_policy.conf**).



Note: Perl modules such as *perl-JSON* and *perl-Sys-Syslog* may need to be installed.

Create an App ID and App Key for the Splunk script on the log server:

```
pbadmin --rest -g SPLUNK-DATA -m SplunkDataAppID
```

Then edit the **/opt/pbul/scripts/closeactionsplunk.pl** script on the log server(s) and change the configurable items appropriately:

```
17
18 my $pbr_appid = "REPLACE-ME";
```

```
19 my $pbr_appkey = "REPLACE-ME";  
20
```

In addition to editing the App ID and App Key, several other edits may be necessary:

The **closeactionsplunk.pl** script currently uses the **auth** syslog facility. Depending on the log server OS, this may need to be changed to **authpriv** in **closeactionsplunk.pl**, or **auth** may need to be configured in addition to **authpriv** in **/etc/syslog.conf** (**rsyslog.conf**, etc.).

The **closeactionsplunk.pl** script uses **/usr/sbin/pbrestcall** internally. This works for installations without a prefix or suffix. If a prefix/suffix installation is used, edit the script to use the appropriate prefix/suffix for **pbrestcall**. The **closeactionsplunk.pl** script uses the default rest port (**24351**), which may need to be changed depending on the actual port used. That port number currently appears in the line:

```
my $pbr_url = https://localhost:24351/REST
```



Tip: The Splunk app can be located at <https://splunkbase.splunk.com/app/4017/> or from within the Splunk GUI under **Apps > Find More Apps**.

REST API for Privilege Management for Unix and Linux

A REST API has been developed for Privilege Management for Unix and Linux to allow other software to configure, customize, and retrieve data from Privilege Management for Unix and Linux. The API is web based and uses industry standard modern components, connectors, and data elements within a distributed and secure enterprise environment. The software is installed on the policy server, log server or run/submit hosts, alongside a suitable HTTP service (one which supports FastCGI), that provides the communications between the client and the REST services.

The REST API provides a RESTful interface for product settings, policy configuration, and I/O log retrieval. The REST API can be used with Privilege Management for Unix and Linux v7.1.0 and later.



For more information, please see "REST API" on page 510.

Multi-Byte Character Set Support

Privilege Management for Unix and Linux uses the locale settings on the host operating systems to support UTF-8 multi-byte character sets in Privilege Management for Unix and Linux policy files, I/O logs, and installation scripts. To correctly use Privilege Management for Unix and Linux in a multi-byte character set environment, you must ensure the following:

- All Privilege Management for Unix and Linux hosts (policy server host, log host, run host, submit host, and so on) have their locale settings correctly configured to the same locale.
- All processes that start at boot time or that are started by **inetd** or **xinetd** inherit the locale settings.



Note: UTF-8 multi-byte character sets are not yet supported in the following Privilege Management for Unix and Linux components:

- shells (**pbsh**, **pbksh**)
- utilities (**pbvi**, **pbnvi**, **pbless**, **pbmg**, **pbumacs**)
- browser interface (**pbgui**)



Note: If the environment variable **LANG**, or one of the environment variables **LC_XXXX** is set to an invalid value, Privilege Management for Unix and Linux components do not error and set **LANG** to **C**. You must ensure **LANG** is correctly set, or if not set correctly, other components of Privilege Management for Unix and Linux (policy server, log server, run host and submit host), are also using **C** or a single-byte character set.

PAM to RADIUS Authentication Module

Starting in v8.5, Privilege Management for Unix and Linux includes a PAM module (**pam_radius_auth**) to support authentication against a configured RADIUS server. The module allows Privilege Management for Unix and Linux to act as a RADIUS client for authentication and accounting requests.

You must have a RADIUS server already installed and configured before using this module. Your RADIUS server must also have the Privilege Management for Unix and Linux host requesting authentication already defined as a RADIUS client.

To configure Privilege Management for Unix and Linux to use **pam_radius_auth**, perform the following steps.

1. Locate the PAM to RADIUS Authentication Module:

Upon installation, the PAM module (**pam_radius_auth**) can be found in **/usr/lib/beyondtrust/pb**. It may be copied to a custom location or the system's default PAM module directory (for example, **/lib/security** or **/usr/lib/security**).

2. Configure the PAM configuration to use **pam_radius_auth**:

Configure a PAM configuration file for **pam_radius_auth** which would define a service stack using the **pam_radius_auth** module. For most Unix operating systems, it can be added in **/etc/pam.conf**. On Linux, it is a separate file in **/etc/pam.d** directory. The service name defined here may be used in the PAM-related Privilege Management for Unix and Linux settings keyword, policy functions, and variables.



Example:

```
/etc/pam.d/pbul_pam_radius:  
#task control module  
auth required pam_radius_auth.so  
account required pam_radius_auth.so  
password required pam_radius_auth.so
```

3. Create/locate the **pam_radius_auth** configuration file:

The **pam_radius_auth** configuration file identifies the RADIUS server(s) that performs the authentication. By default, the **pam_radius_auth** configuration file is **/etc/raddb/server**. You can use a different path/filename and use the module option field in the PAM config file to specify the location:



Example:

```
/etc/pam.d/pbul_pam_radius:  
auth required pam_radius_auth.so conf=<filepathname>
```

4. Set up the **pam_radius_auth** configuration file:

Edit the **pam_radius_auth** configuration file and add a line that represents your RADIUS server using this format:

```
server[:port] shared_secret [timeout]
```

server

Required. RADIUS server name or IP address.

port	Optional. Specify if the port name or number if different from the defined <i>radius</i> port name in /etc/services .
shared_secret	Required. The authentication key defined in the client configuration file for this host on the RADIUS server.
timeout	Optional. The number of seconds the module waits before deciding that the server has failed to respond. The default timeout is 3 seconds .

**Example:**

```
216.27.61.130:1812 secretCnz9CkUtIeHqtCya89LzPTJEq0VnLCNA2SB9KWhIoSnC 10
```

5. Set up Privilege Management for Unix and Linux to use the **pam_radius_auth** module.



For more information on using the services defined here, please see "*Pluggable Authentication Modules*" on page 210.

Privilege Management for Unix and Linux Component, Directory, and File Locations

i For the locations of the Privilege Management for Unix and Linux components, directories, and files, along with other changes and post-installation instructions, please see the [Privilege Management for Unix and Linux Installation Guide](https://www.beyondtrust.com/docs/privilege-management/unix-linux/index.htm) at <https://www.beyondtrust.com/docs/privilege-management/unix-linux/index.htm>.

Sudo Wrapper

After Privilege Management for Unix and Linux is installed and its clients deployed throughout the enterprise, you can ideally start using pbrun instead of sudo to request secured tasks. However, you might need time to modify preexisting scripts or become accustomed to typing **pbrun**. On Linux x86-64 systems, administrators have the option to install and configure a sudo wrapper, a Perl script which facilitates the translation of sudo options into pbrun options and uses pbrun to execute the requested command. This way, users can continue typing **sudo** but pbrun is used to elevated privileges.



IMPORTANT!

Consider the following before installing the sudo wrapper:

- The sudo wrapper is currently supported on Linux x86 64-bit systems only.
- The sudo wrapper and its installation do not touch the preexisting **sudoers** file. The system administrator must migrate the rules from sudoers to the Privilege Management for Unix and Linux policy before installing the sudo wrapper.
- Many of sudo's switches need to be implemented in the policy of Privilege Management for Unix and Linux. This modified policy must be in place prior to installing the sudo wrapper to have those options available.

Packaging

The Perl script **pbsudo-wrapper.pl** is added to the **bin** directory in the TAR file.

Starting with PMUL version 22.1, the sudo wrapper is available only with the Linux distribution: **pmul_linux.x86-64**.

Install Details

pbinstall

The pbinstall program has a new **-O** switch to install sudo wrapper. The Linux host where pbinstall is run must already have an unprefixed/unsuffixed pbrun installed and configured. Before attempting to install sudo wrapper, you must already have an updated Privilege Management for Unix and Linux policy in place that contains the important prerequisites mentioned in the notes above. To ensure that sudo wrapper is installed in the correct environment, the **-O** switch is purposely exclusive and cannot be combined with the other pbinstall options.

When installing the sudo wrapper, pbinstall locates the actual sudo binary and renames it to a backup name (with the suffix **.orig**), after which, the Perl script **pbsudo-wrapper.pl** is copied from the distribution to the same location and renamed **sudo**.

pbuninstall

The pbuninstall program has a new **-O** switch to manually uninstall the sudo wrapper but leave Privilege Management for Unix and Linux components intact.

When uninstalling the sudo wrapper, pbuninstall locates the backup sudo binary (suffixed with **.orig**) and renames it back to its regular name.

Demo Policy Files

Default policy files `<pbuldir>/pbul_policy.conf` and `<pbuldir>/pbul_functions.conf` contain sample instructions that define a **Sudo** role. This **Sudo** role is disabled by default, but it illustrates how you can craft a policy to support the sudo wrapper options.

<policydir>/pbul_policy.conf:

```
# This enables "Sudo role", which allows root (or any user in SudoUsers) to run any command on
the current host (or any host in SudoHosts)
# By default, this role is disabled. To enable this set EnableSudoRole to true below.
#
EnableSudoRole = false;
SudoUsers = {"root"};
SudoHosts = {submithost, TargetSubmitHostShortName};
SudoRole();
```

<policydir>/pbul_functions.conf:

```
## Procedure SudoRole:
## If 'EnableSudoRole' is enabled, it allows any user in SudoUsers list to run any command on
hosts in SudoHosts
##
procedure SudoRole()
{
    if ( EnableSudoRole && user in SudoUsers && (runhost in SudoHosts || TargetRunHostShortName
in SudoHosts) )
    {
        SetRunEnv("root", false);

        if (getenv("SUDOLOGIN") == "true") {
            setenv("SHELL", "!!!");
            setenv("HOME", "!~!");
            runcwd = "!~!";
            runargv[0] = "-" + basename(getenv("SHELL", "/bin/sh"));
            unsetenv("SUDOLOGIN");
            unsetenv("SUDOUSERSHELL");
        }

        if (getenv("SUDOPRESERVE") == "true") {
            setenv("USER", runuser);
            setenv("USERNAME", runuser);
            setenv("LOGNAME", runuser);
            unsetenv("SUDOPRESERVE");
        } else {
            #runcwd = "!~!";
            #setenv("SHELL", "!!!");
            #setenv("HOME", "!~!");
            setenv("USER", runuser);
            setenv("USERNAME", runuser);
            setenv("LOGNAME", runuser);
            setenv("PWD", runcwd);
        }
    }
}
```

```
        setenv("PATH", "/bin:/usr/bin:/usr/local/bin:/sbin:/usr/sbin:/opt/pbis/bin");
        keepenv("SHELL", "HOME", "USER", "USERNAME", "LOGNAME", "PWD", "PATH",
                "TERM", "DISPLAY", "SUDO_GID", "SUDO_UID", "SUDO_USER",
                "SUDO_COMMAND");
    }
    accept;
}
}
```

Settings

The Privilege Management for Unix and Linux settings file contains settings that control its operation. This section describes in detail how to configure these settings.

Base Directory

basedir

- Version 21.1.0 and earlier: **basedir** setting not available.
- Version 22.1.0 and later: **basedir** setting available.

During installation and at runtime, Privilege Management for Unix and Linux creates helper files and directories. Some files/subdirectories are created in the default location `/opt/<prefix>pbul<suffix>`. Consequently, some settings such as **datbasedir**, **licensestatsdb**, and **writequeuepath**, automatically use `/opt/<prefix>pbul<suffix>` as the default path.

Starting with Privilege Management for Unix and Linux v22.1.0, **pbinstall** has an option for administrators to change the default location from `/opt/pbul` and automatically updates and enables the settings that normally use `/opt/pbul` as the parent directory. After installation, the changed path is saved in the **basedir** setting.



For more information, please see **pbinstall** at <https://www.beyondtrust.com/docs/privilege-management/unix-linux/install/programs/index.htm#pb-install>.

Default

```
basedir /opt/<prefix>pbul<suffix>
```

Used on

All hosts

Client and Server Programs

Privilege Management for Unix and Linux is a multi-level client-server application. The client programs are the programs that request services. The server programs are the programs that provide services. Any combination of client or server programs can exist on the same or different hosts.

Privilege Management for Unix and Linux Client Programs

Program	Started From	Runs On
pbguid	browser host	Any host for settings file Policy server host for policy editor

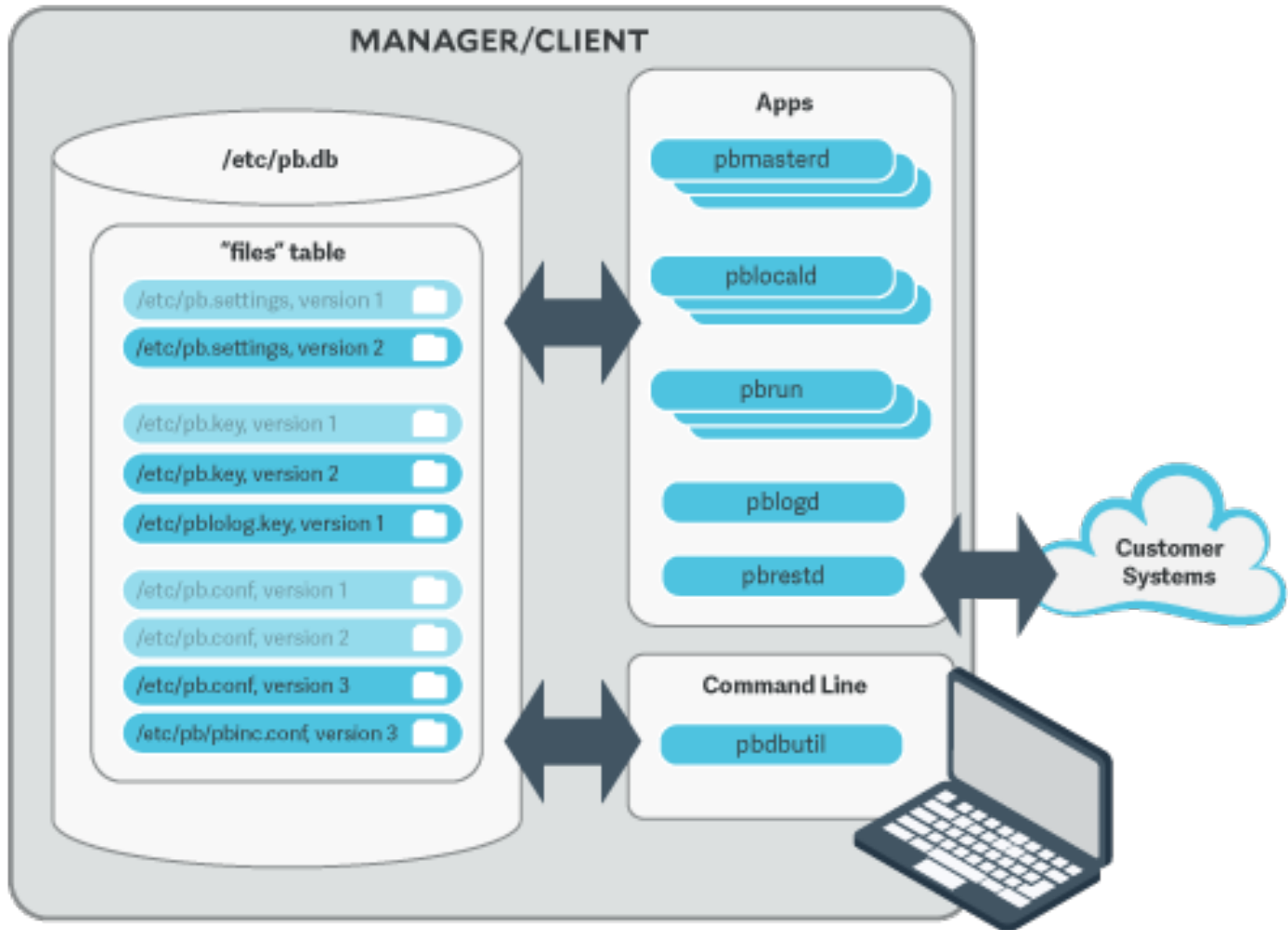
		Log host for event log and I/O log viewers (policy server host if no log host is used).
pbksh	submit host	submit host
pbrun	submit host	submit host
pbsh	submit host	submit host
pbssh	submit host	submit host

Privilege Management for Unix and Linux Server Programs

Program	Started By	Runs On
pblocald	pbmastered	run host
pblogd	pbmastered pblocald Client programs in local mode	log host
pbmastered	Client programs	policy server host

Configuration and Settings Database

The latest version of Privilege Management for Unix and Linux provides new facilities in the area of configuration and settings change management. To provide these facilities the existing configuration files, including the **pb.settings**, policy configuration scripts and encryption keys can now be stored within a database. The database allows the storage of versioning information, and allows the rollback of individual configuration files, or indeed complete sets of files from the command line.



To use the new change management facilities, simply import files into the database use the **pbdbutil**.

Example:

```
# pbdbutil --cfg -i /etc/pb.settings /opt/pbul/policies/pb.conf /etc/pb.key
{"fname":"/etc/pb.settings","version":1}
```



```
# pbdbutil --cfg -i /etc/pb.settings /etc/pb.rest.key
{"fname":"/etc/pb.settings","version":1}
{"fname":"/etc/pb.rest.key","version":1}
```

As soon as the files are imported they are versioned and every Privilege Management for Unix and Linux binary uses the current database copy in preference to the existing files. The existing file is renamed and a comment inserted at the top of the file as an aide-memoire.



Note: Because the database now stores the imported files, we recommend the original files are moved as a backup. The backup will also serve as a reminder that these files no longer represent the current version.

To make changes to the files, export them, make the changes, and re-import the files:

```
# pbdbutil --cfg -e /etc/pb.settings
{"fname":"/etc/pb.settings","version":"#"}
# vi /etc/pb.settings
# pbdbutil --cfg -i /etc/pb.settings
{"fname":"/etc/pb.settings","version":2}
```

The version is incremented, and the current version stored.

Once the administrator is happy with a set of configuration files, these files may all be tagged together at their current version. This allows an administrator to retrieve all the tagged files at their respective versions at a later date. To tag a group of files, use **pbdbutil --cfg -t**:

```
# pbdbutil --cfg -l
{"version":1,"pathname":"/opt/pbul/policies/pb.conf","deleted":0,"created":"2018-05-21 14:26:51"}
{"version":2,"pathname":"/etc/pb.settings","deleted":0,"created":"2018-05-21 14:26:48"}

# pbdbutil --cfg -t "working" /etc/pb.settings /opt/pbul/policies/pb.conf
{"fname":"/etc/pb.settings","tag":"working","version":"#"}
{"fname":"/opt/pbul/policies/pb.conf","tag":"working","version":"#"}
```

The administrator can then export the files as a group:

```
# pbdbutil --cfg -l
{"version":1,"pathname":"/opt/pbul/policies/pb.conf","deleted":0,"created":"2018-05-21 14:26:51"}
{"version":2,"pathname":"/etc/pb.settings","deleted":0,"created":"2018-05-21 14:26:48"}

# pbdbutil --cfg -l /etc/pb.settings
{"pathname":"/etc/pb.settings","version":1,"tag":null,"deleted":0,"created ":"2018-05-21
14:26:46"}
{"pathname":"/etc/pb.settings","version":2,"tag":"working","deleted":0,"created":"2018-05-21
14:26:48"}

# pbdbutil --cfg -e -V working
{"fname":"/etc/pb.conf","version":"#"}
{"fname":"/etc/pb.settings","version":"#"}
```

i For more information on other options to import, export, tag and diff database configuration files, please see "[pbdbutil](#), [pbadmin](#)" on page 378.

Database Settings

databasedir

- **Version 9.3.0 and earlier:** `databasedir` setting not available.
- **Version 9.4.0 and later:** `databasedir` setting available.

The `databasedir` setting is used to specify the default path to databases used within the product, that are specified with a relative path.



Example:

```
databasedir /opt/pbul/dbs
servicedb pbsvc.db
```

The example above configures the service database to be `/opt/pbul/dbs/pbsvc.db`. However,

```
databasedir /opt/pbul/dbs
servicedb /etc/pbsvc.db
```

does not affect **`servicedb`**

Default

```
databasedir /opt/<prefix>pbul<suffix>/dbs
```

Used On

All hosts

databaselocktimeouts

- **Version 10.0.1 and earlier:** `databaselocktimeouts` setting not available.
- **Version 10.1.0 and later:** `databaselocktimeouts` setting available.

The `databaselocktimeouts` setting is used to specify the database timeout values for all the various services with Privilege Management for Unix and Linux. Busy systems can sometimes produce delays in processing data because of contention for database resources. Usually a service will timeout on the specific database and return an error. However, this may not always be the best course of action.

The current list of services include:

- **license**
- **rns**
- **dbsync**
- **akapolicy**
- **iologidx**
- **restkey**
- **fim**
- **event**
- **logcache**
- **rbp**
- **sudo**
- **sched**
- **polpvar**
- **logarchive**
- **intprod**
- **clientreg**
- **pbpolicy**

There is also a default service that is applied when a more specific setting has not been configured.

Values take the format of **service=<delay>,<retries>**, where **delay** is in microseconds.



Example:

```
databaselocktimeouts default=1000,30 fim=2000,60 rbp=500,10
```

Default

No default value

Used On

All hosts

lockfileexpiry

Configuration and settings files are centrally managed by Privilege Management for Unix and Linux using a database. Administrators are allowed to lock the file(s) contained in databases to manage concurrency. The **lockfileexpiry** setting specifies the length of time (in seconds) before that file lock expires.

**Example:**

```
lockfileexpiry 45
```

Default

```
0
```

Client Registration

The Client Registration feature facilitates the installation and configuration of new Privilege Management for Unix and Linux clients into the enterprise. It consists of a centralized Registration Profile service, normally found on the primary server. This service is configured with customized profiles that match the settings required for the installation of hosts that provide differing roles in the organization. When new Privilege Management for Unix and Linux clients are installed these profiles are retrieved, providing the configuration required to complete the installation.

cIntregdb

- **Version 9.3.0 and earlier:** `cIntregdb` setting not available.
- **Version 9.4.0 and later:** `cIntregdb` setting available.

The `cIntregdb` option specifies the path to the Client Registration database. This file is created in `basedir` by default, unless the file name starts with `/`.

**Example:**

```
cIntregdb /etc/pbclntreg.db
```

Default

```
cIntregdb /opt/<prefix>pbul<suffix>/dbs/pbregclnt.db
```


Used On

Primary servers

Example Client Registration Procedure

1. Install the primary license server, making sure that the REST services are installed. Make a note of the port number of the REST service. Also make sure that if the host has any firewalls installed, that this TCP port is unblocked.

2. After main installation is complete check that the REST service is running (`ps -ef | grep pbconfigd`), and if it isn't, follow normal procedures to start the service.
3. Create a REST application ID and key.

 **Example:**

```
pdbutil --rest -g appid  
{"appkey": "934bbab5-503e-4c40-8486-90c748142431"}
```

4. Ensure the `pb.settings` file has all the necessary configuration, including encryption and encryption keys defined.
5. Now we are ready for a client install. Run the normal `pbinstall` on the client and when asked whether to use Client Registration, answer **yes**, and provide responses to the Client Registration configuration questions.

 **Example:**

```
Do you wish to utilize Client Registration? [yes]  
  
Enter the Application ID generated on the Primary License Server: appid  
Enter the Application Key generated on the Primary License Server: 934bbab5-503e-4c40-  
8486-90c748142431  
Enter the Primary License Server address/domain name for registering clients:  
pbmasterhost.mydomain.com  
Enter the Primary License Server REST TCP/IP port [24351]: 24351  
Enter the Registration Client Profile name [default]: default
```

6. `pbinstall` then defaults all the configuration of the install to the retrieved `pb.settings` and transfers all the necessary keyfiles automatically.

Role Based Policy

Role Based Policy has been implemented to simplify the definition of policy for administrators. Policies are kept within structured records in a database, simplifying maintenance, decreasing system load, increasing throughput, and providing a comprehensive REST API to integrate policy management with existing customer systems and procedures, including simplified bulk import/export of data. Once the customers' data is held within the Role Based Policy database it is much easier to provide management information, such as user entitlement reports. The policy data is grouped into users, hosts, commands, time/dates, and roles detailed in the schema below.

Database Schema

User Groups

User Groups define groups of users and/or wildcard patterns that match usernames:

```
CREATE TABLE usergrp (  
  id INTEGER PRIMARY,  
  name TEXT,  
  description TEXT,  
  disabled INTEGER CHECK(disabled BETWEEN 0 AND 1), -- 0=enabled, 1=disabled  
  type CHAR(1) CHECK (type IN ('I','E')), -- I=Internal, E=external  
  single INTEGER CHECK(disabled BETWEEN 0 AND 1),  
  extinfo TEXT, -- external lookup info  
  UNIQUE {name, id});  
  
CREATE TABLE userlist (  
  id INTEGER  
  user TEXT, -- "glob" wildcard  
  PRIMARY KEY(id,user),  
  FOREIGN KEY(id) REFERENCES usergrp(id) DEFERRABLE INITIALLY DEFERRED);
```

Each user group has multiple user list entries that specify names and/or wildcards that match both submit and run user names when matched by the role.

There is additionally a special value user list entry that allows the Role Based Policy to match on the specified submit username, that is **\$submituser\$**.

USER GROUP

1. User Group ID (key, unique)
 2. User Group Name
 3. User Group Description
 4. Disabled
 5. Single (single user/group)
 6. Group Type (internal/external)
 7. External Group connection info (encoded - json ?)
- } Composite Key

MEMBERSHIP LIST

1. User Group ID (foreign key)
 2. Member String (glob/regex)
- } Composite Key

Host Groups

Host groups define groups of hosts and/or wildcard patterns that match hostnames:

```
CREATE TABLE hostgrp ( id INTEGER PRIMARY, name TEXT UNIQUE,
description TEXT,
disabled INTEGER CHECK(disabled BETWEEN 0 AND 1), -- 0=enabled, 1=disabled
type CHAR(1) CHECK (type IN ('I','E')), -- I=Internal, E=external
extinfo TEXT -- external lookup info

CREATE TABLE hostlist (
id INTEGER REFERENCES hostgrp(id), host TEXT, -- "glob" wildcard
PRIMARY KEY(id,host)
);
```

Each host group has multiple host list entries that specify names and/or wildcards that will match both submit and run host names when matched by the role.

HOST GROUP

1. Host Group ID (key, unique)
2. Host Group Name (unique)
3. Host Group Description
4. Disabled
5. Group Type (Internal/external)
6. External Group connection info (encoded - json ?)

HOST LIST

- | | | |
|---|---|---------------|
| <ol style="list-style-type: none"> 1. Host Group ID (foreign key) 2. Host String (glob/regex) | } | Composite Key |
|---|---|---------------|

Command Groups

Command groups define groups of commands and/or wildcard patterns that match commands:

```
CREATE TABLE cmdgrp (
  id INTEGER PRIMARY, name TEXT UNIQUE,
  description TEXT,
  disabled INTEGER CHECK(disabled BETWEEN 0 AND 1) -- 0=enabled, 1=disabled
);

CREATE TABLE cmdlist (
  id INTEGER REFERENCES cmdgrp(id),
  cmd TEXT, -- "glob" wildcard
  rewrite TEXT, -- new command (see below)
  PRIMARY KEY(id,cmd)
);
```

Each command group has multiple command list entries that specify commands and/or wildcards that will match the submitted command name when matched by the role, and a rewrite column to rewrite the command that will be executed. The rewrite is in a similar format to Bourne/Bash shell arguments. For example, **\$0**, **\$1**, **etc**, **\$*** and **\$#**. Rewrite uses the original command to substitute arguments into the new rewritten command.

COMMAND GROUP

1. Command Group ID (key, unique)
2. Command Group Name (unique)
3. Command Group Description
4. Disabled

COMMAND LIST

1. Command Group ID (foreign key)
 2. Command String (glob/regex)
 3. Command re-write
- } Composite Key

Time/Date Groups

Time/date groups define groups of times/dates and/or wildcard patterns that match times/dates:

```
CREATE TABLE tmdategrp (
  id INTEGER PRIMARY,
  name TEXT UNIQUE,
  description TEXT,
  disabled INTEGER CHECK(disabled BETWEEN 0 AND 1)  -- 0=enabled, 1=disabled

CREATE TABLE tmdatelist (
  id INTEGER REFERENCES tmdategrp(id),
  tmdate TEXT,                                     -- json format - see below
  PRIMARY KEY(id,tmdate)
);
```

Each time/date group has multiple time/date list entries that specify times/dates and/or wildcards that match the submitted command name when matched by the role, and a rewrite column to rewrite the command that is executed. Each individual time/date is specified in JSON format, and can be one of two different formats:

- **From/To specific date range:** from and to are specified in epoch seconds:

```
'{ "range" : { "from" : 1415851283, "to": 1415887283 } }'
```

- **Day of the Week:** each day is specified as an array of hours. Each hour is a number representing 15 minute intervals defined as a binary mask:

```
1 1 1 1
^ 0 to 14 minutes of the hour
^-- 15 to 29 minutes of the hour
```

```

^---- 30 to 44 minutes of the hour
^----- 45 to 59 minutes of the hour Therefore the values range from 0 to 15:

' {
"mon" : [0,0,0,0,0,0,0,15,15,15,15,15,15,15,15,15,15,3,0,0,0,0,0,0],
"tue" : [0,0,0,0,0,0,0,15,15,15,15,15,15,15,15,15,15,3,0,0,0,0,0,0],
"wed" : [0,0,0,0,0,0,0,15,15,15,15,15,15,15,15,15,15,3,0,0,0,0,0,0],
"thu" : [0,0,0,0,0,0,0,15,15,15,15,15,15,15,15,15,15,3,0,0,0,0,0,0],
"fri" : [0,0,0,0,0,0,0,15,15,15,15,15,15,15,15,15,15,3,0,0,0,0,0,0],
"sat" : [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
"sun" : [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
}'
    
```

TIME/DATE GROUP

1. Time/Date Group ID (key, unique)
2. Time/Date Group Name (unique)
3. Time/Date Group Description
4. Disabled

TIME/DATE LIST

1. Time/Date Group ID (foreign key)
 2. Time/Date/Day (json encoded)
- } Composite Key

Roles

Roles are the entities that tie all the other entities together to define a role.

```

CREATE TABLE role (
  id INTEGER PRIMARY KEY,
  name TEXT UNIQUE,
  rorder INTEGER,                -- rule order for matching
  description TEXT,
  disabled INTEGER CHECK(disabled BETWEEN 0 AND 1), -- 0=enabled, 1=disabled
  risk INTEGER CHECK(risk >= 0),
  action CHAR(1) CHECK (action IN ('A','R')),      -- A=Accept, R=Reject
  iolog TEXT,          -- iolog template
  script TEXT,         -- pbparse script
  tag TEXT,            -- Arbitrary tag that will allow grouping of roles
  comment TEXT,        -- Arbitrary comment field that can contain anything
  message TEXT,        -- Accept/reject message (templated)
  variables TEXT,      -- Contains JSON formatted Policy Script variables to set (templated)
  varmatch TEXT,       -- Contains JSON formatted Policy Script variables to match
  auth TEXT,           -- Contains JSON formatted array of authentication methods (templated)
  rpt INTEGER DEFAULT 1 -- 1=on, 0=off, include Role in Entitlement Report
);

CREATE TABLE roleusers (
    
```



```
id INTEGER REFERENCES role(id),
users INTEGER REFERENCES usergrp(id),
type CHAR(1) CHECK (type IN ('S','R')),          -- S=Submit, R=Run User
PRIMARY KEY (id,users,type)
);

CREATE TABLE rolehosts (
id INTEGER REFERENCES role(id),
hosts INTEGER REFERENCES hostgrp(id),
type CHAR(1) CHECK (type IN ('S','R')),          -- S=Submit, R=Run Host
PRIMARY KEY (id,hosts,type)
);

CREATE TABLE rolecmds (
id INTEGER REFERENCES role(id),
cmds INTEGER REFERENCES cmdgrp(id),
PRIMARY KEY (id,cmds)
);

CREATE TABLE roletmdates (
id INTEGER REFERENCES role(id),
tmdates INTEGER REFERENCES tmdategrp(id),
PRIMARY KEY (id,tmdates)
);
```

Each role has multiple users, hosts, commands and time/dates. When the Policy Engine matches against roles, complete records are selected from the database as fully populated roles, sorted by the role attribute **rorder**. Once the first record has been matched, the attributes of the role are applied to the session, and the Policy Engine either accepts or rejects the session. The iolog template is the normal script format log file, for example `/var/log/io_log.XXXXXX`. The script is a full Privilege Management for Unix and Linux script that is called if the role has been accepted. This script can carry out extra processing to authorize the session (and can therefore override the accept/reject status with an implicit command), and can carry out extended environment configuration as would normal Privilege Management for Unix and Linux script.

ROLE

1. Role Group ID (key, unique)
2. Role Name (unique)
3. Role Order
4. Role Description
5. Disabled
6. Risk
7. Accept/Reject
8. I/O Log Template
9. Script
10. Tg
11. Comment
12. Message Template
13. Policy Script Variables
14. Varmatch
15. Authentication Methods
16. Include In Entitlement Report

SUBMIT/RUN USER LIST

1. Role ID (foreign key)
 2. User Group ID (foreign key)
 3. Type (submit/run)
- } Composite Key

COMMAND LIST

1. Role ID (foreign key)
 2. Command Group ID (foreign key)
- } Composite Key

SUBMIT/RUN HOST LIST

1. Role ID (foreign key)
 2. Host Group ID (foreign key)
 3. Type (submit/run)
- } Composite Key

TIME/DATE LIST

1. Role ID (foreign key)
 2. Time/Date Group ID (foreign key)
- } Composite Key

Change Management Events

There are two different approaches to maintaining the Role Based Policy database. The first, simple method is to access the tables using **pbdbutil** at the command line. Each change is individual, instantaneous, and *live* immediately. Although for smaller organizations this is adequate, larger organizations have a more controlled procedural access method.

Role Based Policy database *change transactions* can be enabled using the settings **rbptransactions**. Once enabled, before changes can be made, the administrator must begin a change transaction, specifying a reason why the change is being made. This is logged and the whole Role Based Policy database is then locked for update. Only that administrator can continue to make changes. These changes will NOT be mirrored in the *live* authorization process and can continue to be made by that administrator alone, and when completed can be committed or rolled back. Once the changes are committed they are all applied to the database as one update, and a change management event is generated. If the changes are rolled back, they are discarded and nothing changes.



Note: *If a change transaction is begun, and the administrator leaves it open and fails to close the transaction, any other administrator with access can force the rollback of the changes. This requires specifying a reason, and logs a change management event. The change transactions are necessary once the GUI policy updates are implemented to force database integrity.*

To enable the logging of change management events, each client needs to configure the **pb.setting changemanagementevents yes**, log servers need to define **eventdb <path>**, and the REST **pbconfigd** service must be running.

Role Based Policy Settings

The following settings are used and need to be set when Role Based Policy and change management is implemented and used.

policydb

- **Version 8.5 and earlier:** `policydb` setting not available.
- **Version 9.0 and later:** `policydb` setting available.

The `policydb` setting specifies the path to the Role Based Policy database. If the value is not an absolute path (does not start with “/”), the database file is created in the location provided by `datbasedir` setting.

Default

```
/opt/<prefix>pbul<suffix>/dbs/pbrbpolicy.db
```

Used On

Policy server hosts

rolebasedpolicy

- **Version 8.5 and earlier:** `rolebasedpolicy` setting not available.
- **Version 9.0 and later:** `rolebasedpolicy` setting available.

Enable or disable Role Based Policy checking. The default is no.



Example:

```
rolebasedpolicy  yes
```

Default

```
rolebasedpolicy  no
```

Used On

Policy server hosts

rbptransactions

- **Version 8.5 and earlier:** `rbptransactions` setting not available.
- **Version 9.0 and later:** `rbptransactions` setting available.

Enable the use of Role Based Policy Transactions to ensure integrity.



Example:

```
rbptransactions    yes
```

Default

```
rbptransactions    no
```

Used On

Policy server hosts

changemanagementevents

- **Version 8.5 and earlier:** `changemanagementevents` setting not available.
- **Version 9.0 and later:** `changemanagementevents` setting available.

Enable/Disable the logging of Change Management Events when maintaining databases.

A logserver must be installed before enabling the `changemanagementevents` keyword.



Example:

```
changemanagementevents    yes
```

Default

```
changemanagementevents    no
```

The following settings are also used in Role Based Policy.

- `eventdb <path>`

The path to the Change Management Event Database. The default is `/opt/pbul/dbs/pbevent.db`.

- **pbresturi** <string>

The partial REST url string between the hostname and **/REST**. There is no default for this setting.

- **pbrestport** <port#>

The REST port. Default value is the **base port + 6**.

- **eventdestinations** <taxonomy>=<remote>,<db>,<authvt>,<syslog>,</path/to/file>,</path/to/executable>
<taxonomy>=.....

Events can be sent to a remote host (primary logserver), event database, syslog, a *flat file* on the file system, or a binary or script for consumption. Multiple destinations can be specified for each taxonomy with commas to separate. The default destination is **authvt=db**.

- **eventformats** <taxonomy>=<csv|json> <taxonomy>=....

Specify the format of events to log (except database records which are always JSON). The default format is **JSON**

- **pbresttimeskew** <num>

The maximum time in seconds that hosts are mismatched by (we recommend that the customer use a time synchronization service). The default is **60 seconds**.



For more information, please see:

- ["eventdb" on page 123](#)
- ["pbresturi" on page 270](#)
- ["pbrestport" on page 272](#)
- ["eventdestinations" on page 123](#)
- ["eventformats" on page 125](#)
- ["pbresttimeskew" on page 273](#)

Role Based Policy Entitlement Reports

Privilege Management for Unix and Linux v10.1.0 introduced Role Based Policy Entitlement Reports. These reports are available to the user from the **pbrun** command using **-e**, or to the administrator as an overall report using **pbdbutil --rbp -R**. To provide a comprehensive report on what users can access commands on which hosts, and when they are allowed to run them.

pbdbutil: Role Based Policy Options

The **pbdbutil** Role Based Policy options introduced in Privilege Management for Unix and Linux v10.1.0 are described below.

```
pbdbutil --rbp [<options>] [ <file> <file> ...]
-R { json param } Report user entitlements from the database
-R Add option to display commands
-R Add option to display time/date restrictions
-R Add option to display additional role options
-E { json param } List user entitlements data from the database
where { json param } is one or more of:
"submituser" : "user1" Specify submit user or wildcard
"submithost" : "host1" Specify submit host or wildcard
"runuser" : "user1" Specify run user or wildcard
"runhost" : "host1" Specify run host or wildcard
"command" : "command" Specify command or wildcard
```

pbrun Options

Privilege Management for Unix and Linux v10.1.0 introduced the following options that are available only when Role Based Policy is enabled:

pbrun -e	Return the entitlement report for the current user at level 1.
pbrun -e 1	Return the entitlement report for the current user at level 1.
pbrun -e 4	Return the entitlement report for the current user at level 4.
pbrun --entitlement=4	Return the entitlement report for the current user at level 4.

Examples of Entitlement Reports



Example: Level 1 Report

```
=====
Privilege Management for Unix and Linux Role Based Policy Entitlement Report - Level 1
-----
Date/Time: 2018-06-18 09:07:23
User: root
Belongs to the following Roles:
Admin
```



```

=====
Role Order: 1
Name: Admin
Description: Super users and admins
Action: allowed
Tag:
Membership: Admins
Submit Host(s): Any PBUL Host
Run Host(s): Any PBUL Host
Commands may be executed as user(s): root,admin,user*
Please use the '-u' flag to select user at run time.
eg: pbrun -u runuser command [arguments]
User may request the following commands using pbrun:
/bin/find *,/usr/bin/ls,/bin/ls,/bin/cat *,/bin/ls *,/usr/bin/ls *,/usr/bin/rm *,
/usr/bin/cat *,/usr/bin/find *,/sbin/shutdown *,/bin/more *,/bin/id,/usr/bin/more *,
/usr/bin/mount *,/bin/ln *,/bin/mount *,/bin/rm *,/usr/sbin/shutdown *,
/usr/bin/ln *,/usr/bin/id,/sbin/ifconfig *,/usr/sbin/ifconfig *
    
```



Example: Level 2 Report

```

=====
Privilege Management for Unix and Linux Role Based Policy Entitlement Report - Level 2
-----
Date/Time: 2018-06-18 09:07:28
User: root
Belongs to the following Roles:
Admin
=====
Role Order: 1
Name: Admin
Description: Super users and admins
Action: allowed
Tag:
Risk: 1
Membership: Admins
Submit Host(s): Any PBUL Host
Run Host(s): Any PBUL Host
Commands may be executed as user(s): root,admin,user*
Please use the '-u' flag to select user at run time.
eg: pbrun -u runuser command [arguments]
User may request the following commands using pbrun:
Command Group: User Commands
Description: Common UNIX Commands
/bin/ls executes: /bin/ls
/bin/ls * executes: /bin/ls *
/usr/bin/ls executes: /usr/bin/ls
/usr/bin/ls * executes: /usr/bin/ls *
/bin/cat * executes: /bin/cat *
/usr/bin/cat * executes: /usr/bin/cat *
/bin/find * executes: /bin/find *
/usr/bin/find * executes: /usr/bin/find *
    
```




```

/bin/more * executes: /bin/more *
/usr/bin/more * executes: /usr/bin/more *
/bin/rm * executes: /bin/rm -i $*
/usr/bin/rm * executes: /usr/bin/rm -i $*
/bin/ln * executes: /bin/ln *
/usr/bin/ln * executes: /usr/bin/ln *
/bin/id executes: /bin/id
/usr/bin/id executes: /usr/bin/id
Command Group: Admin Commands
Description: Common Superuser Commands
/sbin/shutdown * executes: /sbin/shutdown *
/usr/sbin/shutdown * executes: /usr/sbin/shutdown *
/bin/mount * executes: /bin/mount *
/usr/bin/mount * executes: /usr/bin/mount *
/sbin/ifconfig * executes: /sbin/ifconfig *
/usr/sbin/ifconfig * executes: /usr/sbin/ifconfig *
    
```



Example: Level 3 Report

```

=====
Privilege Management for Unix and Linux Role Based Policy Entitlement Report - Level 3
-----
Date/Time: 2018-06-18 09:07:30
User: root
Belongs to the following Roles:
Admin
=====
Role Order: 1
Name: Admin
Description: Super users and admins
Action: allowed
Tag:
Risk: 1
Membership: Admins
Submit Host(s): Any PBUL Host
Run Host(s): Any PBUL Host
Commands may be executed as user(s): root,admin,user*
Please use the '-u' flag to select user at run time.
eg: pbrun -u runuser command [arguments]
User may request the following commands using pbrun:
Command Group: User Commands
Description: Common UNIX Commands
/bin/ls executes: /bin/ls
/bin/ls * executes: /bin/ls *
/usr/bin/ls executes: /usr/bin/ls
/usr/bin/ls * executes: /usr/bin/ls *
/bin/cat * executes: /bin/cat *
/usr/bin/cat * executes: /usr/bin/cat *
/bin/find * executes: /bin/find *
/usr/bin/find * executes: /usr/bin/find *
    
```



```

/bin/more * executes: /bin/more *
/usr/bin/more * executes: /usr/bin/more *
/bin/rm * executes: /bin/rm -i $*
/usr/bin/rm * executes: /usr/bin/rm -i $*
/bin/ln * executes: /bin/ln *
/usr/bin/ln * executes: /usr/bin/ln *
/bin/id executes: /bin/id
/usr/bin/id executes: /usr/bin/id
Command Group: Admin Commands
Description: Common Superuser Commands
/sbin/shutdown * executes: /sbin/shutdown *
/usr/sbin/shutdown * executes: /usr/sbin/shutdown *
/bin/mount * executes: /bin/mount *
/usr/bin/mount * executes: /usr/bin/mount *
/sbin/ifconfig * executes: /sbin/ifconfig *
/usr/sbin/ifconfig * executes: /usr/sbin/ifconfig *
Date and Time restrictions for Role 'Admin':
Time/Date Group: Any Time
Description: Any Time
Monday: 01:00am to 12:14pm
Tuesday: 01:00am to 12:14pm
Wednesday: 01:00am to 12:14pm
Thursday: 01:00am to 12:14pm
Friday: 01:00am to 12:14pm
Saturday: 01:00am to 12:14pm
Sunday: 01:00am to 12:14pm
    
```



Example: Level 4 Report

```

=====
Privilege Management for Unix and Linux Role Based Policy Entitlement Report - Level 4
-----
Date/Time: 2018-06-18 09:07:32
User: root
Belongs to the following Roles:
Admin
=====
Role Order: 1
Name: Admin
Description: Super users and admins
Action: allowed
Tag:
Risk: 1
Membership: Admins
Submit Host(s): Any PBUL Host
Run Host(s): Any PBUL Host
Commands may be executed as user(s): root,admin,user*
Please use the '-u' flag to select user at run time.
eg: pbrun -u runuser command [arguments]
User may request the following commands using pbrun:
    
```



```

Command Group: User Commands
Description: Common UNIX Commands
/bin/ls executes: /bin/ls
/bin/ls * executes: /bin/ls *
/usr/bin/ls executes: /usr/bin/ls
/usr/bin/ls * executes: /usr/bin/ls *
/bin/cat * executes: /bin/cat *
/usr/bin/cat * executes: /usr/bin/cat *
/bin/find * executes: /bin/find *
/usr/bin/find * executes: /usr/bin/find *
/bin/more * executes: /bin/more *
/usr/bin/more * executes: /usr/bin/more *
/bin/rm * executes: /bin/rm -i $*
/usr/bin/rm * executes: /usr/bin/rm -i $*
/bin/ln * executes: /bin/ln *
/usr/bin/ln * executes: /usr/bin/ln *
/bin/id executes: /bin/id
/usr/bin/id executes: /usr/bin/id
Command Group: Admin Commands
Description: Common Superuser Commands
/sbin/shutdown * executes: /sbin/shutdown *
/usr/sbin/shutdown * executes: /usr/sbin/shutdown *
/bin/mount * executes: /bin/mount *
/usr/bin/mount * executes: /usr/bin/mount *
/sbin/ifconfig * executes: /sbin/ifconfig *
/usr/sbin/ifconfig * executes: /usr/sbin/ifconfig *
Date and Time restrictions for Role 'Admin':
Time/Date Group: Any Time
Description: Any Time
Monday: 01:00am to 12:14pm
Tuesday: 01:00am to 12:14pm
Wednesday: 01:00am to 12:14pm
Thursday: 01:00am to 12:14pm
Friday: 01:00am to 12:14pm
Saturday: 01:00am to 12:14pm
Sunday: 01:00am to 12:14pm
Additional Role Options:
Additional Authentication Required: no
Session Recording Enabled: yes
Extended Script Policy: no
Custom accept/reject message: no
    
```



Example: Level 1 Report with Command Filter

```

pbdbutil -P --rbp -R '{ "command":"/usr/bin/*"}'
=====
Privilege Management for Unix and Linux Role Based Policy Entitlement Report - Level 1
-----
Date/Time: 2018-06-18 09:09:10
User: *
    
```



```

Belongs to the following Roles:
Admin,users
=====
Role Order: 1
Name: Admin
Description: Super users and admins
Action: allowed
Tag:
Risk: 1
Membership: Admins
Submit Host(s): Any PBUL Host
Run Host(s): Any PBUL Host
Commands may be executed as user(s): root,admin,user*
Please use the '-u' flag to select user at run time.
eg: pbrun -u runuser command [arguments]
User may request the following commands using pbrun:
/usr/bin/ls,/usr/bin/mount *,/usr/bin/ls *,/usr/bin/cat *,/usr/bin/find *,
/usr/bin/rm *,/usr/bin/ln *,/usr/bin/more *,/usr/bin/id
=====
Role Order: 4
Name: users
Description: Normal users
Action: allowed
Tag:
Membership: Users
Submit Host(s): nfs.company.com,build.company.com,staging.company.com
Run Host(s): nfs.company.com,build.company.com,staging.company.com
Commands will execute as user: user*
User may request the following commands using pbrun:
/usr/bin/ls,/usr/bin/ls *,/usr/bin/find *,/usr/bin/cat *,/usr/bin/ln *,
/usr/bin/rm *,/usr/bin/more *,/usr/bin/id
    
```



Example: Level 4 Report with Command Filter

```

=====
Privilege Management for Unix and Linux Role Based Policy Entitlement Report - Level 4
-----
Date/Time: 2018-06-18 09:09:26
User: *
Belongs to the following Roles:
Admin,users
=====
Role Order: 1
Name: Admin
Description: Super users and admins
Action: allowed
Tag:
Risk: 1
Membership: Admins
Submit Host(s): Any PBUL Host
    
```



```

Run Host(s): Any PBUL Host
Commands may be executed as user(s): root,admin,user*
Please use the '-u' flag to select user at run time.
eg: pbrun -u runuser command [arguments]
User may request the following commands using pbrun:
Command Group: Admin Commands
Description: Common Superuser Commands
/usr/bin/mount * executes: /usr/bin/mount *
Command Group: User Commands
Description: Common UNIX Commands
/usr/bin/ls executes: /usr/bin/ls
/usr/bin/ls * executes: /usr/bin/ls *
/usr/bin/cat * executes: /usr/bin/cat *
/usr/bin/find * executes: /usr/bin/find *
/usr/bin/more * executes: /usr/bin/more *
/usr/bin/rm * executes: /usr/bin/rm -i $*
/usr/bin/ln * executes: /usr/bin/ln *
/usr/bin/id executes: /usr/bin/id
Date and Time restrictions for Role 'Admin':
Time/Date Group: Any Time
Description: Any Time
Monday: 01:00am to 12:14pm
Tuesday: 01:00am to 12:14pm
Wednesday: 01:00am to 12:14pm
Thursday: 01:00am to 12:14pm
Friday: 01:00am to 12:14pm
Saturday: 01:00am to 12:14pm
Sunday: 01:00am to 12:14pm
Additional Role Options:
Additional Authentication Required: no
Session Recording Enabled: yes
Extended Script Policy: no
Custom accept/reject message: no
=====
Role Order: 4
Name: users
Description: Normal users
Action: allowed
Tag:
Risk: 1
Membership: Users
Submit Host(s): build.company.com,nfs.company.com,staging.company.com
Run Host(s): build.company.com,nfs.company.com,staging.company.com
Commands will execute as user: user*
User may request the following commands using pbrun:
Command Group: User Commands
Description: Common UNIX Commands
/usr/bin/ls executes: /usr/bin/ls
/usr/bin/ls * executes: /usr/bin/ls *
/usr/bin/cat * executes: /usr/bin/cat *
/usr/bin/find * executes: /usr/bin/find *
/usr/bin/more * executes: /usr/bin/more *
    
```



```
/usr/bin/rm * executes: /usr/bin/rm -i $*
/usr/bin/ln * executes: /usr/bin/ln *
/usr/bin/id executes: /usr/bin/id
Date and Time restrictions for Role 'users':
Time/Date Group: Working Week
Description: Working Week
Monday: 01:00am to 12:14pm
Tuesday: 01:00am to 12:14pm
Wednesday: 01:00am to 12:14pm
Thursday: 01:00am to 12:14pm
Friday: 01:00am to 12:14pm
Saturday: none
Sunday: none
Additional Role Options:
Additional Authentication Required: no
Session Recording Enabled: no
Extended Script Policy: no
Custom accept/reject message: no
```

Settings and Configuration Policy File Names

It is possible for one machine to have multiple installations of Privilege Management for Unix and Linux. This can be done to balance resources or partition usage.

For example, if the prefix is **g** and the suffix is **7**, the **pblogd** executable is installed as **gpblogd7**. The same prefixes and/or suffixes also apply to all of the files that are used by default for Privilege Management for Unix and Linux. Continuing the example, the settings file changes from **pb.settings** to **gpb.settings7**. The same prefix and/or suffix is also prepended and/or appended to **pb.key**, **pb.conf**, and all the binary files in this installation set.

This feature can be used to allow for multiple configurations with different functionality, such as Kerberized and non-Kerberized installations on the same machine. For example, a suffix of **k** could be used for the Kerberized files.

Throughout this guide there are references to **/etc/pb.settings** and **/opt/pbul/policies/pb.conf**. This usage assumes that the default settings and configuration files are used.

Privilege Management for Unix and Linux programs recognize prefixes and/or suffixes on their names and use settings and configuration files with the same prefixes and/or suffixes. This feature allows multiple differing configurations on the same machine. This feature is consistent with the rest of the product where any number of settings files may be used. Examples of the settings and configuration file names are listed in the following table:

Default Settings and Configuration Policy File Names

Program Name	Settings Filename	Default Policy Configuration File
pbrun	/etc/pb.settings	/opt/pbul/policies/pb.conf
pbrun_alt	/etc/pb.settings_alt	/opt/pbul_alt/policies/pb.conf_alt
my.pnrun	/etc/my.pnpb.settings	/opt/my.pnpbul/opt/my.pbpb.conf
kpbmasterd	/etc/kpb.settings	/opt/kpbul/opt/kpb.conf
kpblocald.2	/etc/kpb.settings.2	/opt/kpbul.2/opt/kpb.conf.2

If the prefix or suffix cannot be determined, then **/etc/pb.settings** and **/opt/pbul/policies/pb.conf** are used.

File Locations

environmentfile

- **Version 5.2 and earlier:** **environmentfile** setting not available.
- **Version 6.0 and later:** **environmentfile** setting available.

The **environmentfile** setting enables you to specify the absolute path and file name of an environment file. Privilege Management for Unix and Linux can incorporate the environment variables that are specified in the environment file into the run environment. These environment variables are applied on the run host after the Accept event has been logged.

The **environmentfile** setting can be overridden in the policy by setting the **runenvironmentfile** variable.

The environment file must consist of the following:

- Comment lines, which have a **#** character in the first non-whitespace position
- Blank lines
- Bourne shell compatible environment variable setting lines with the form **NAME=VALUE**



Note: Each line in the file must contain less than 1024 characters. Line continuation is not supported. This file must not contain any shell commands or constructs other than the setting of environment variables. Comments must not appear on the same line as an environment variable.



Note: This keyword does not apply to **pbssh**. If it is present in the settings file, it does not have any effect on **pbssh** and is ignored.



Example:

```
environmentfile "/etc/environment"
environmentfile "/etc/environment2"
```

Default

No default value

Used on

Run hosts



For more information, please see "runenvironmentfile" in the *Privilege Management for Unix and Linux Policy Language Guide* at <https://www.beyondtrust.com/docs/privilege-management/unix-linux/index.htm>.

pbrunpath

- **Version 6.0 and earlier:** **pbrunpath** setting not available.
- **Version 6.2.5 and later:** **pbrunpath** setting available.

The **pbrunpath** setting specifies the absolute path to the directory that contains the **pbrun** and **pbssh** executable files on the **pbguid** host. This setting enables the Task Manager feature to work properly.



Example:

```
pbrunpath /usr/bin/
```

Default

No default value

Used on

- **pbguid** host
- Run hosts

policydir and policyfile

- **Version 4.0.0 and later:** **policydir** and **policyfile** settings available.

Privilege Management for Unix and Linux uses the settings file **/etc/pb.settings** (or a prefixed and/or suffixed settings file). By default, the main policy file is **/opt/pbul/policies/pb.conf** (**/etc/pb.conf** prior to v9.4.3). You can choose a different configuration policy file by using the **policydir** and **policyfile** settings.

The **policyfile** setting contains the name of a policy file. If the name is an absolute path (starting with a **/**), then that file name is used. If the **policyfile** setting does not start with a **/**, then the value of **policydir** is prepended to it.



Example:

```
# Use /etc/my.pb.conf as the configuration policy file
policyfile /etc/my.pb.conf

# Use /usr/local/powerbroker/powerbroker.conf as the configuration
# policy file
policydir /usr/local/powerbroker
policyfile powerbroker.conf
```

policydir also controls include files in the configuration policy. If a file name in an include statement starts with a **/**, then that file is used. However, if the file name does not start with a **/**, the **policydir** setting is prepended to it.



Example: If `policydir` is set to `/etc/pb.includes`:

```
include "admin.conf";  
# Includes /etc/pb.includes/admin.conf
```

but

```
include "/etc/admin.conf";  
# Includes /etc/admin.conf
```

Defaults

```
policydir /opt/<prefix>pbul<suffix>/policies (/etc prior to v9.4.3)  
policyfile /opt/<prefix>pbul<suffix>/policies/<prefix>pb.conf<suffix>
```

Used on

Policy server hosts

polycypersistentvariabledb

- **Version 9.4.4 and earlier:** `polycypersistentvariabledb` setting not available.
- **Version 9.4.5 and later:** `polycypersistentvariabledb` setting available.

The `polycypersistentvariabledb` option specifies the path to the Persistent Variable database used in script policy.



Example:

```
polycypersistentvariabledb /etc/pbpersist.db
```

Default

```
polycypersistentvariabledb /opt/<prefix>pbul<suffix>/dbs/pbpolpersistvar.db
```

Used On

Policy servers

tempfilepath

- **Version 10.2.0 and earlier:** `tempfilepath` setting not available.
- **Version 10.3.0 and later:** `tempfilepath` setting available.

The `tempfilepath` option defines a temporary path to use as the temporary filesystem for PMUL binaries. The value can be any valid directory.

Default

```
/tmp
```

Used On

Policy servers

lockfilepath

- **Version 10.2.0 and earlier:** `lockfilepath` setting not available.
- **Version 10.3.0 and later:** `lockfilepath` setting available.

The `lockfilepath` option defines a lock file path for PMUL binaries, when needed. The value can be any valid secure directory.

Default

```
lockfilepath /opt/<prefix>pbul<suffix>/locks
```

Used On

Policy servers

lockfiletimeout

Privilege Management for Unix and Linux uses a lockfile when it writes into flat files (e.g. policy file or settings file) on the filesystem. This prevents multiple processes overwriting each other's work. The `lockfiletimeout` setting specifies the maximum number of seconds for a process to wait for a lock to be free.



Example:

```
lockfiletimeout 45
```

Default

```
lockfiletimeout 30
```

Host and Port Specifications

In several settings, Privilege Management for Unix and Linux requests hosts, ports, or both.

Port Specifications

A port can consist of a file name for Unix/Linux Domain Sockets or a number or service name with an optional interface. The port numbers for Privilege Management daemons must use the non-reserved system ports. The allowed port numbers are **1024** to **65535** (inclusive).

The optional interface specification is useful on a machine that has multiple interfaces or network cards. The interface specification enables you to specify the exact interface to use.

Format

```
[port=]<number or service name>[:interface=<host name or IP address>] [port=]<absolute file name>
```

The optional interface may be a host name or IP address. If the optional interface is specified for an incoming port, then only connections on that interface are accepted. If the optional interface is specified for an outgoing port, then only that interface is used to form the connection.

Port Format	Description	Example
[port=]<absolute file name>	A local (Unix/Linux Domain) socket	/tmp/pbport port=/tmp/pbmasterd_port
[port=]<number>	A numeric TCP/IP port	24345 port=24345
[port=]<service name>	A service name from /etc/services or NIS	pbmasterd port=pbmasterd
[port=]<number>: interface=<hostname>	A numeric TCP/IP port on a specific interface	24345:interface= localhost port=24345: interface= host.domain
[port=]<service name>:interface=<IP address>	A service name from /etc/services or NIS on a specific interface	pbmasterd: interface= 127.0.0.1 port=pbmasterd: interface= 192.168.1.1

When using a symbolic name, Privilege Management services need to be defined in **/etc/services**. Doing so enables the superdaemon, Privilege Management for Unix and Linux programs, and system utilities (such as **nslookup**) to know the symbolic names of the ports (and services) that are provided by Privilege Management.

Host Specifications

Hosts may be a host name, an IP address, a DNS SRV lookup entry, the **_auto** DNS SRV lookup, or a path to an external program. If appropriate, they may also be followed by a port specification.

Format

```
<host name or IP address>
```

A valid host name can contain any letter or digit. It can also contain a hyphen. The last character must be a letter or a digit.

```
<DNS SRV lookup>
```



Note: *Privilege Management for Unix and Linux validates data returned from DNS SRV lookups to exclude any data that contains the following illegal characters:*

```
/'~?!@#$$%^&*(){}<>=,|\\\"
```

When Privilege Management for Unix and Linux components processing the **submitmasters**, **altsubmitmasters**, **acceptmasters**, and **logservers** keywords encounter an entry that begins with an underscore (`_`) those components perform a DNS query. That query returns a port value for the service, as well as a list of hosts providing that service. A complete DNS SRV lookup specification includes the service name, protocol (which is always `_tcp`), domain, and a port specification. The **submitmasters**, **acceptmasters**, and **logservers** keywords have the ability to specify a lookup in the format:

```
_<pbul service name>._tcp.<domain name>[:port=<port>[:interface=<IP or hostname>]]
```

All elements are optional except the service name. If the domain name portion is missing, it is up to the resolver and its configuration to define and use a search order (for example, `/etc/resolv.confsearch` entry). If the `._tcp` portion is missing, it is added automatically. This implementation supports the SRV hostname, port and priority values, and ignores the SRV weight value.

If the port value returned in the DNS query is non-zero, it overrides the corresponding port specified in **pb.settings** (for example, a DNS lookup for **submitmasters**, which returns a port `> 0`, overrides any port specified in the **submitmasters** keyword and in the **masterport** keyword.

If the port specified in DNS is zero, the port specified in the corresponding **submitmasters**, **altsubmitmasters**, or **logservers** keyword (if specified there) is used. If not specified, the port specified in the corresponding **masterport**, **logport** keywords is used.

If an interface is specified in **pb.settings** (for example, specified in the **submitmasters** keyword or in the **masterport** keyword), that interface is used even if the corresponding port is overridden by the DNS lookup.

The DNS SRV lookup can be specified in any order in the list of hosts, and is processed in that order.



Example:

```
submitmasters host1 _pbmasterd host2
```

*The **pb.settings** entry results in the attempt to use **host1** as a policy server. Only if that fails, DNS SRV records are queried for the **_pbmasterd._tcp** entry. Only if all hosts specified in the DNS SRV query results fail, **host2** is contacted.*

Once an SRV lookup takes place, the priority values specified in DNS is used for those entries.



Note: *DNS can be configured for round-robin or randomized results for a minimal form of load balancing, if desired.*

In this case, the order of DNS results are modified by DNS and Privilege Management for Unix and Linux simply processes the SRV query results in the order received.

The **randomizesubmitmasters** keyword should not be used with the use of DNS SRV lookups.



Example:

```
submitmasters host1 _pbmasterd host2
```



If the **pb.settings** entry is as shown above, the **randomizesubmitmasters** may cause **host2** to be contacted prior to performing the **_pbmasterd** service lookup, prior to **host1**. The **randomizesubmitmasters** does not affect the order of processing the SRV query results. This would conflict with the SRV priority values.



Note: When a DNS SRV lookup is specified for the **acceptmasters** keyword, the SRV priority and port are irrelevant.



Note: **pbsyncd** uses the **logservers** keyword to identify logservers, however this uses a different port than **pblogd**, so any DNS SRV records for host providing **pbsyncd** should have the port set to **0**.

```
<_auto DNS SRV lookup>
```

The special service name **_auto** is used to build a service name from the Privilege Management installation's prefix and suffix.

When **_auto** is used for **submitmasters** and **acceptmasters**, the resulting service name is:

```
_<prefix>pbmasterd<suffix>
```

When **_auto** is used for **altsubmitmasters** the resulting service name is:

```
_<prefix>altpbmasterd<suffix>
```

When **_auto** is used for **logservers** the resulting service name is:

```
_<prefix>pblogd<suffix>
```

The full **_auto** specification is:

```
_auto[:port=1.2.3.4[:interface=<IP or hostname>]]
```



Note: Use of **_auto** for **submitmasters** and **acceptmasters** results in the same SRV lookup. Use caution when using **_auto** for **altsubmitmasters**, as the runhost's **acceptmasters** list must include the hosts specified by the **altsubmitmasters** SRV lookup.

```
<`/path/to/external/program`>
```

The **submitmasters**, **altsubmitmasters**, **acceptmasters**, and **logservers** keywords support a mechanism to execute an external program that returns a single value. The external program path and filename should be contained within backticks without whitespace. Command line arguments to the external program are not supported. Redirection and backgrounding the external program are not supported.

Output of the external program should be a single valid host entry (for example, a hostname, an IP address, a file name, a netgroup, or an SRV lookup; including the optional **[:port=<port>[:interface=<interface>]]** specification) for the supported keywords, whitespace (other than a trailing newline) is only acceptable when specifying a program and its arguments such as: **/path/to/pbmasterd -ar**.



Note: The external program must be writable only by root. The external program must execute within 2 seconds, or the value of `masterprotocoltimeout` converted to seconds, whichever is greater. When an external program does not complete within the specified timeframe, the program may become a "zombie" process.

The optional port/interface specification can be output by the external program, or immediately following the backticks specifying the external program but not both.

Errors that occur evaluating the external program are logged to the appropriate log file, however they will not be displayed to the submituser's display. Thus the submitting user sees a generic error message such as `3366 Invalid submitmasters setting`.



Example:

```
myhost.mydomain
  _auto
  _pbmasters
  _pbmasters._tcp.mydomain.
  _pbmasters._tcp. mydomain.:port=12345
`/bin/get_first_submitmaster`
```


Submit Task Requests to a Policy Server Daemon

Submit hosts need to know how to connect to a policy server daemon and how long to wait for failovers. The connections are defined by the **submitmasters** and **masterport** settings. Connection timing is controlled by the **masterdelay** and **masterprotocoltimeout** settings.

Policy Server Connection Settings

submitmasters

- **Version 4.0.0 and later:** **submitmasters** setting available

The **submitmasters** setting provides a list of outgoing connection information for submitting task requests to Privilege Management for Unix and Linux policy server daemons.

The list can contain:

- Host names
- A single asterisk (*) denoting a Registry Name Service lookup
- Netgroups in the form:

```
+@name
```

- Hosts to exclude in the form:

```
-name
```

- Netgroups to exclude in the form:

```
-@name
```

- Absolute path names of a local **pbmasterd**. If spaces are required, then the string must be enclosed in quotation marks.
- DNS SRV lookups, in the form:

```
_pbul service name>._tcp.<domain name>[:port=<port>[:interface=<IP or hostname>]]
```

- External programs, in the form:

```
`/path/to/external/program`
```

The following are tried in sequence to determine the port value:

1. The non-zero port value from a DNS SRV lookup
2. The value specified within the **submitmasters** setting
3. The value of the **masterport** setting
4. The **pbmasterd** entry in services

5. Port **24345**
6. The non-zero port value from a DNS SRV lookup
7. The value specified within the **submitmasters** setting
8. The value of the **masterport** setting
9. The **pbmasterd** entry in services
10. Port **24345**


Example:

```
submitmasters myhost.mydomain
submitmasters sparky spot submitmasters host1 host2
submitmasters +@submitmasters -@badmasters -badhost
submitmasters sparky spot "/usr/sbin/pbmasterd -ar"
submitmasters _auto
submitmasters _pbmasters
submitmasters _pbmasters._tcp.mydomain.
submitmasters _pbmasters._tcp. mydomain.:port=12345
submitmasters ` /bin/get_first_submitmaster `
```

Default

No default value

Used on

Submit hosts

altsubmitmasters

- **Version 6.0 and earlier:** **altsubmitmasters** setting not available.
- **Version 6.2.5 and later:** **altsubmitmasters** setting available.

The **altsubmitmasters** setting enables you to control which policy server host a request is sent to, based on the user and the command.

Syntax


```
altsubmitmasters user-list : command-list : host-list
```

where:

- **user-list** is a space-separated list of user names.
- **command-list** is a space-separated list of commands.
- **host-list** is a space-separated list of hosts that follows the same rules as the host list in the **submitmasters** setting.

The lists are separated from each other by the character string " : " (that is, a space, a colon, and a space, without the quotation marks). You can specify multiple **altsubmitmasters** statements in the same settings file.

 **Note:** If you specify one or more policy server hosts in the **altsubmitmasters** list, be sure to include them in the run host's **acceptmasters** list as well.

 **Note:** Caution should be used when using **_auto** for **altsubmitmasters**, as the runhost's **acceptmasters** list must include the hosts specified by the **altsubmitmasters** SRV lookup.

The **user-list** and **command-list** can include regular expression syntax such as *****, **?**, **[**, and **]**.

 **Example:**

```
altsubmitmasters root : ls rm : host1 host2
altsubmitmasters m* : p* : host3
altsubmitmasters user1 user2: ps : host4
submitmasters host5
altsubmitmasters user1 user2: ps : _altmasters._tcp.mydomain.
```

In this example, the submit host sends any **ls** or **rm** requests from **root** to the hosts **host1** or **host2**, any command that starts with the letter **p** from any user name that starts with the letter **m** to the host **host3**, any **ps** command issues by users **user1** and **user2** to **host4**, and any other requests to the host **host5** (using the **submithosts** setting).

Default

No default value

Used On

Submit hosts

masterport

- **Version 4.0.0 and later:** **masterport** setting available.

The port numbers for Privilege Management daemons must use the non-reserved system ports. The allowed port numbers are **1024** to **65535** (inclusive).

 The value for this setting follows the guidelines in "[Port Specifications](#)" on page 77.

 **Example:**

```
masterport 12345
```

**Example:**

```
masterport pbmasterd
```

Default

```
masterport 24345
```

Used on

- Policy Server hosts
- Submit hosts

randomizesubmitmasters

- **Version 6.0 and earlier:** **randomizesubmitmasters** setting not available.
- **Version 6.2.5 and later:** **randomizesubmitmasters** setting available.

The **randomizesubmitmasters** setting forces the submit host to choose a policy server host at random, rather than choosing the first policy server host that is specified in the **submitmasters** setting. This feature balances the load among multiple policy server hosts. System administrators must ensure that all policy server hosts are using the same version of the policy.



Note: The **randomizesubmitmasters** keyword should not be used with the use of DNS SRV lookups.

**Example:**

```
randomizesubmitmasters yes
```

Default

```
randomizesubmitmasters yes
```

Used on

Submit hosts

Receive Task Requests from a Policy Server Daemon

Receiving task requests from a policy server daemon, run hosts need to know which policy server daemon to acknowledge. This is controlled by the **acceptmasters** settings control. Further authentication is possible using the **validatemasterhostname** setting.

acceptmasters

- **Version 4.0.0 and later:** **acceptmasters** setting available.

The **acceptmasters** setting specifies incoming connections from the policy server daemon that Privilege Management for Unix and Linux programs acknowledge.



Note: The policy server hosts in the run host's **acceptmasters** list must also be specified in the submit host's **submitmasters** or **altsubmitmasters** lists.

The list can contain:

- Host names
- A single asterisk (*)denoting a Registry Name Service lookup
- Netgroups in the form:

```
+@name
```

- Hosts to exclude in the form:

```
-name
```

- Netgroups to exclude in the form:

```
-@name
```

- DNS SRV lookups, in the form:

```
_<pbul service name>._tcp.<domain name>.[:port=<port>[:interface=<IP or hostname>]]
```

- External Programs, in the form:

```
`/path/to/external/program`
```

The order of precedence for the **acceptmasters** rules is:

1. Command line for **pblocald -m** or **--accept_masters** argument
2. Setting for **acceptmasters**
3. Netgroup for **pbacceptmasters**



Note: This keyword does not apply to **pbssh**. If it is present in the settings file, it does not have any effect on **pbssh** and is ignored.



Example:

```
acceptmasters myhost.mydomain
acceptmasters sparky spot
acceptmasters +@pbacceptmasters
acceptmasters +@pbacceptmasters -@badmasters -badhost
```

Default

No default value

Used on

Run hosts

allowruntimeoverride

- **Version 5.2 and earlier:** allowruntimeoverride setting not available.
- **Version 6.0 and later:** allowruntimeoverride setting available.

The **allowruntimeoverride** setting allows a runhost's **pb.settings** to override a **runtimeout** value set in the master policy. Each runhost wanting to take advantage of this ability would then set the **runtimeout** keyword in their own **pb.settings**. **allowruntimeoverride** must be set to **yes** to allow this override to occur.



Example:

```
allowruntimeoverride yes
```

Default

```
allowruntimeoverride no
```

Used on


Policy servers

i For more information, please see "runtimeout" in the [Privilege Management for Unix and Linux Policy Language Guide](https://www.beyondtrust.com/docs/privilege-management/unix-linux/index.htm) at <https://www.beyondtrust.com/docs/privilege-management/unix-linux/index.htm>.

runtimeout

- **Version 4.0.0 and later:** runtimeout setting available.

When the policy server allows **runtimeout** overrides, the **runtimeout** keyword is used to set an idle time limit for all secured tasks on this runhost. The **runtimeout** variable specifies the amount of idle time, in seconds, that the submitting user is allowed before the run host terminates the current request.

 **Note:** The **runtimeout** keyword is not honored in local mode or **pbssh**.

The policy server's **runtimeoutoverride** keyword must be set to **yes** to allow this override to occur.

 **Example:**

```
runtimeout 600
```

Default

```
runtimeout 0
```

Used on

Run hosts

See Also

i For more information, please see "runtimeout" in the [Privilege Management for Unix and Linux Policy Language Guide](https://www.beyondtrust.com/docs/privilege-management/unix-linux/index.htm) at <https://www.beyondtrust.com/docs/privilege-management/unix-linux/index.htm>.

Failover

Privilege Management for Unix and Linux can be configured to provide backup user request processing and activity logging. This capability is referred to as failover.

To configure backup request processing, specify the backup policy server hosts by using the **submitmasters** keyword in `/etc/pb.settings` on the submit host.

To configure backup logging, specify the backup log hosts by using the **logservers** keyword in `/etc/pb.settings` on the policy server host.



For more information on failover, please see the following:

- ["submitmasters" on page 81](#)
- ["logservers" on page 99](#)

Fine Tuning Policy Server and Failover Connection Timing

masterdelay

- **Version 4.0.0 and later:** **masterdelay** setting available.

When a request is submitted, the policy server hosts that are listed in the **submitmasters** line are tried in the order they appear, from left to right. The **masterdelay** setting enables the administrator to adjust the amount of time between failover attempts.

Without a specified time-out, the client tries the first policy server host on the **submitmasters** line. If it does not receive a response within 500 milliseconds, then the client adds the second policy server host. If neither responds in the next 500 milliseconds, then the client adds the third policy server host, and so on. By specifying a **masterdelay**, you can change the 500 millisecond waiting period before the client goes on to the next policy server host.

With a **masterdelay** of 0 milliseconds, you get the fastest possible connection, but the policy server you connect to may not be predictable. You might also increase network traffic, depending on the number of connections that are opened.

With a larger **masterdelay**, you can increase the predictability, but you might also increase the time needed to form a failover connection. The longer the delay, the more predictable the sequence is.



Example:

```
masterdelay 200
```

Default

```
masterdelay 500
```

Used on

Submit hosts

masterprotocoltimeout

- **Version 4.0.0 and later:** `masterprotocoltimeout` setting available.

After a connection is established, the programs perform some protocol checks to verify a proper and working connection. Some types of protocol failures could take a long time to determine (for example, wrong service running on the policy server port, or mismatched encryption types/keys).

The `masterprotocoltimeout` setting enables the administrator to control the maximum time to wait for protocol completion. If a protocol step does not complete within the specified number of milliseconds, then the client continues to try the next policy server host in sequence. A value of `-1` indicates no protocol timeout.



Example:

```
masterprotocoltimeout 2000
```

Default

```
masterprotocoltimeout 500
```

Used on

- Policy server hosts
- Run hosts
- Submit hosts

Fine Tuning Log Servers and Failover Connection Timing

logserverdelay

- **Version 4.0.0 and later:** `logserverdelay` setting available.

When a log request is processed, the log servers that are listed in the `logservers` line are tried in the order they appear, from left to right. The `logserverdelay` setting enables the administrator to adjust the amount of time between failover attempts.

Without a specified time-out, the logging program (for example, `pbrun`, `pbrunmasterd`, `pbrunlocald`, etc.) tries the first log server on the `logservers` line. If it does not receive a response within 500 milliseconds, then it adds the second log host. If neither responds in the next 500 milliseconds, then it adds the third log host, and so on. By specifying a `logserverdelay`, you can change the 500 millisecond waiting period before the logging program goes on to the next log server.

With a `logserverdelay` of 0 milliseconds, you get the fastest possible connection, but the log server that you connect to may not be predictable. You might also increase network traffic, depending on the number of connections that are opened.

With a larger `logserverdelay` you can increase the predictability, but you might also increase the time needed to form a failover connection. The longer the delay, the more predictable the sequence is.

**Example:**

```
logserverdelay 2500
```

Default

```
logserverdelay 500
```

Used on

- Policy server hosts
- Run hosts
- Submit hosts by **pbksh** and **pbsh** when a policy server is not available

logserverprotocoltimeout

- **Version 4.0.0 and later:** **logserverprotocoltimeout** setting available.

After a connection is established, the programs perform some protocol checks to verify a proper and working connection. Some types of protocol failures can take a very long time to determine. For example, the wrong service running on the log server port, or mismatched encryption types/keys.

The **logserverprotocoltimeout** setting enables the administrator to control the maximum time to wait for protocol completion. If a protocol step does not complete within the specified number of milliseconds, then the logging program continues to try the next log server in sequence. A value of **-1** indicates no protocol timeout.

If the **iologack** setting is used, then the **logserverprotocoltimeout** setting also controls how long a submit host should wait for an acknowledgment from the log host.

**Example:**

```
logserverprotocoltimeout 2000
```


Default

```
logserverprotocoltimeout 500
```

Used on

- Log hosts
- Policy server hosts


- Run hosts
- Submit hosts by **pbksh** and **pbsh** when a policy server host is not available


 For more information, please see "*iologack*" on page 140.


randomizelogservers

- **Version 9.2.0 and earlier:** **randomizelogservers** setting not available.
- **Version 9.3.0 and later:** **randomizelogservers** setting available.

The **randomizelogservers** setting forces the policy server/submit host/run host to choose a log server host at random, rather than choosing the first available log server host that is specified in the **logservers** setting. This feature balances the load among multiple log server hosts.

 **Note:** The use of **randomizelogservers** can cause accept and finish events to be located on different log servers if the log servers are configured with **eventdestinations** set to a flat file (**authvt=<file>**) or an SQLite Database (**authvt=db**). However, if **eventdestinations** is set to **authvt=<DSN>** (same ODBC Oracle or MySQL database on all the log servers), then the accept and finish events are stored on the same Oracle or MySQL server. The default **randomizelogservers** setting is **no**.

 **Note:** The **randomizelogservers** keyword should not be used with the use of DNS SRV lookups. The **randomizelogservers** keyword can result in accept and finish events logged on different logservers, causing the need to merge iologs.

 **Example:**

```
randomizelogservers yes
```

Default

```
randomizelogservers no
```

Used on

- Submit hosts
- Run hosts
- Policy servers

Acknowledge Failovers

transparentfailover

- **Version 5.1.1 and earlier:** `transparentfailover` setting not available.
- **Version 5.1.2 and later:** `transparentfailover` setting available.

A **transparentfailover** occurs when an initial connection to a policy server or logserver host has failed and the program performs a failover to another available policy server or logserver host in the list. To acknowledge that a user failover has occurred, error messages from the failed connection are displayed to the user.

The **transparentfailover** setting enables you to suppress the following failover error messages:

- Any Kerberos initialization error
- *3084 initMangle failure during startup*
- *3089 Could not send initial protocol header to Policy Server*
- *3090 Did not receive initial protocol header from Policy Server*
- *8534 Policy Server on %s is not SSL enabled*
- *1913 Invalid Policy Server daemon on Policy Server host %s*

When **transparentfailover** is set to **yes**, failover error messages listed above are suppressed. To display failover error messages, set **transparentfailover** to **no** in the `pb.settings` file.



Example:

```
transparentfailover yes
```

Default

```
transparentfailover yes
```

Used on

Submit hosts

Policy Server Host Connections to plocald

When a policy server host accepts a request, it tries to connect to **plocald** on the run host. This connection is controlled by the **localport** setting for TCP/IP connections or by the **plocaldcommand** setting for local connections.

localport

- **Version 4.0.0 and later:** **localport** setting available.

The value of the **localport** setting is used in two cases:

- For a policy server host, this setting specifies the well-known port for **plocald**.
- For a standalone local daemon, if the port is not specified on the command line, then the value of this setting is used.



Note: The value of this setting follows the guidelines in "Port Specifications" on page 77.

The port numbers for Privilege Management for Unix and Linux daemons must use the non-reserved system ports. The allowed port numbers are **1024** to **65535** (inclusive).

The following order is used to determine the port for **plocald**:

1. The value of **localport**
2. The **plocald** entry in services
3. Port **24346**



Note: This keyword does not apply to **pbssh**. If it is present in the settings file, it does not have any effect on **pbssh** and is ignored.



Example:

```
localport 12345
```

```
localport plocald
```

Default

```
localport 24346
```

Used on

- Policy server hosts
- Run hosts

pblocaldcommand

- **Version 4.0.0 and later:** `pblocaldcommand` setting available.

If the policy server host and the run host are the same machine, then you can specify the path and arguments to `pblocald`. Doing so eliminates the overhead of forming a connection between `pbmasterd` and `pblocald`. If the command line contains spaces, then you must enclose the entire command line in quotation marks.



Note: This keyword does not apply to `pbssh`. If it is present in the settings file, it does not have any effect on `pbssh` and is ignored.



Example:

```
pblocaldcommand "/usr/sbin/pblocald -s"
```

Default

No default value

Used on

Policy server hosts

Connections to SSH-Managed Devices

Using the **pbssh** program and the Privilege Management for Unix and Linux policy, you can control who can access SSH-managed devices (such as a Windows computer or certain network devices) and what commands users can execute on those devices.

The **pbssh** binary enables you to access and manage third party devices that are not accessible using the traditional **pbrun** binary. These devices can be a router, a Windows machine, a Unix/Linux server where Privilege Management is not installed, or any other appliance that can be managed by SSH.

Using the **pbssh** program and the Privilege Management for Unix and Linux policy, you can now control which users can access these SSH-managed devices and which commands the designated users can execute on these devices without having Privilege Management installed on the devices. As long as SSH is configured properly to access the device, **pbssh** is able to access and manage the device. In addition, the logging features (I/O logging and event logging) are available for **pbssh**.

Compared to the traditional Privilege Management **pbrun** program, there are a few limitations on what **pbssh** can achieve. Because Privilege Management is not installed on the target device (**runhost**), all run variables in the Privilege Management policy are not applicable to **pbssh**. Therefore **pbssh** cannot elevate the privileges of a user on the target device. Using the function **setkeystrokeaction**, you can limit the commands that a user can execute on the target device. However, you cannot allow the user to run a command on that device that they are normally not able to run.

When invoked, the **pbssh** program connects to the target host (specified with the required field **-h**) using an existing user account (defined by the required **-u** option) on the target machine. The target host will likely require a password. If Password Safe (**pkrun**) is available and configured properly (using Privilege Management settings **pkrunfile**, **pk_cert**, and **pk_servers**) the password is automatically retrieved from the Password Safe server. Otherwise the user is prompted to provide the password.

The following settings govern this feature:

pbsshlog

- **Version 6.0 and earlier:** **pbsshlog** setting not available.
- **Version 6.2.5 and later:** **pbsshlog** setting available.

pbsshlog contains the name of the **pbssh** diagnostic log file.



Example:

```
pbsshlog /var/log/pbssh.log
```

Default

No default value

Used on

Submit hosts

pbsshshell

- **Version 6.0 and earlier:** `pbsshshell` setting not available.
- **Version 6.2.5 and later:** `pbsshshell` setting available.

The `pbsshshell` setting specifies the shell to be used while connected to an SSH-managed device. Privilege Management for Unix and Linux uses this setting to correctly perform I/O logging.



Example:

```
pbsshshell bash
```

Default

```
pbsshshell /bin/sh
```

Used on

Submit hosts

pk_cert

- **Version 6.0 and earlier:** `pk_cert` setting not available.
- **Version 6.2.5 and later:** `pk_cert` setting available.

The `pk_cert` setting specifies the absolute path to the Password Safe certificate to use when using `pbssh`.



Example:

```
pk_cert /etc/mypk.cert
```

Default

No default value

Used on

Submit hosts



For more information, please see "[pbssh](#)" on page 468.

pk_servers

- **Version 6.0 and earlier:** `pk_servers` setting not available.
- **Version 6.2.5 and later:** `pk_servers` setting available.

The `pk_servers` setting specifies one or more Password Safe servers (by host name or IP address) from which to obtain the password to use when logging in to an SSH-managed device.



Example:

```
pk_servers host0 192.168.1.125
```

Default

No default value

Used on

Submit hosts

pkrunfile

- **Version 6.0 and earlier:** `pkrunfile` setting not available.
- **Version 6.2.5 and later:** `pkrunfile` setting available.

The `pkrunpath` setting specifies the absolute path and file name of the `pkrun` executable file. This setting enables you to use Password Safe when using `pbssh`.



Example:

```
pkrunfile /usr/bin/pkrun
```

Default

No default value

Used on

Submit hosts



For more information, please see "`pbssh`" on page 468.

pktimeout

- **Version 6.0 and earlier:** `pktimeout` setting not available.
- **Version 6.2.5 and later:** `pktimeout` setting available.

The `pktimeout` setting specifies the amount of time (in seconds) that the `pbssh` program waits for a response from Password Safe. If you specify a value less than 60 seconds, then 60 seconds is used.



Example:

```
pktimeout 100
```

Default

```
pktimeout 60
```

Used on

Submit hosts

shortnamespk

- **Version 6.0 and earlier:** `shortnamespk` setting not available.
- **Version 6.2.5 and later:** `shortnamespk` setting available.

The `shortnamespk` setting enables `pbssh` to connect to a Password Safe host using a short host name instead of a fully-qualified domain name. Specifying **yes** for the `shortnamespk` setting enables short host names; specifying **no** requires that host names be fully-qualified domain names.



Example:

```
shortnamespk yes
```

Default

```
shortnamespk no
```

Used on

Submit hosts

Connections to Log Servers

Hosts that access Privilege Management log servers need to know how to connect to log servers and how long to wait for failovers. The connections are defined by the **logservers** and **logport** settings. These settings must be defined on the policy server host. Run hosts usually obtain this information from the policy server host's policy file and do not need these settings. Log servers need to know only the port.

Connection timing is controlled by the **logserverdelay** and **logserverprotocoltimeout** settings.

logservers

- **Version 4.0.0 and later:** **logservers** setting available.

The **logservers** setting provides a list of outgoing connection information for Privilege Management for Unix and Linux programs that use log servers.

The list can contain:

- Host names
- A single asterisk (*)denoting a Registry Name Service lookup
- Netgroups in the form:

```
+@name
```

- Hosts to exclude in the form:

```
-name
```

- Netgroups to exclude in the form:

```
-@name
```

- Absolute path names of a local **pblogd**. If spaces are required, the string must be quoted.
- DNS SRV lookups, in the form:

```
_<pbul service name>._tcp.<domain name>[:port=<port>[:interface=<IP or hostname>]]
```

- External Programs, in the form:

```
`/path/to/external/program`
```

The following are tried in sequence to determine the port value:

1. The non-zero port value from a DNS SRV lookup
2. The value specified within the **logservers** setting
3. The value of the **logport** setting

4. The **pblogd** entry in services 5.
5. Port **24347**


Example:

```
logservers mylogserver.mydomain
logservers sparky spot
logservers loghost1 loghost2
logservers +@logservers -@badlogservers -badlogserver
logservers sparky spot "/usr/sbin/pblogd"
logservers _auto
logservers _pbmasters
logservers _pbmasters._tcp.mydomain.
logservers _pbmasters._tcp.mydomain.:port=12345
logservers `/bin/get_first_submitmaster`
```

Default

No default value

Used on

- Policy server hosts
- Submit hosts by **pbksh** and **pbsh** when a Policy Server is not available

logport

- **Version 4.0.0 and later:** **logport** setting available.

The port numbers for Privilege Management daemons must use the non-reserved system ports. The allowed port numbers are **1024** to **65535** (inclusive).



Note: The value of this setting follows the guidelines in "[Port Specifications](#)" on page 77.


Example:

```
logport 12345
logport pblogd
```

Default

```
logport 24347
```

Used on

- Log hosts
- Policy server hosts
- Run host
- Submit hosts by **pbksh** and **pbsh** when a policy server host is not available

Host Name Verification

namerresolutiontimeout

- **Version 5.1.1 and earlier:** `nameresolutiontimeout` setting not available.
- **Version 5.1.2 and later:** `nameresolutiontimeout` setting available.

Privilege Management for Unix and Linux attempts to obtain fully qualified domain names when a `pblogd`, `pbksh`, `pbsh`, `pblocald`, `pbrun`, or `pbrun` session is started. The `nameresolutiontimeout` setting defines the timeout period, in seconds, to be used for the request to expire. The timeout period is an approximate time for the daemon (`pbrun`, `pblocald`, `pblogd`) to time out.

Setting `nameresolutiontimeout` to `0` disables this feature. The allowed timeout values range from `1` to `7200` seconds.



Example:

```
nameresolutiontimeout 30
```

Default

```
nameresolutiontimeout 0
```

Used on

- `pblocald`, `pbksh`, `pbsh`, `pbrun`
- `pblogd`, `pbrun`

shortnamesok

- **Version 4.0.0 and later:** `shortnamesok` setting available.

Privilege Management for Unix and Linux attempts to obtain fully qualified domain names for all hosts that are involved in a task request. If some of the name services return fully qualified names but others return only short names, you can instruct Privilege Management for Unix and Linux to check only the short name by setting `shortnamesok` to `yes`.



IMPORTANT!

This option is intended as a temporary measure for discrepancies in a system's name services. This option decreases security because it permits a partial name match (for example, two machines in different domains with the same short name would match). It is preferable to make the name services consistent whenever possible.

**Example:**

```
shortnamesok yes
```

Default

```
shortnamesok no
```

Used on

- Log hosts
- Policy server hosts
- Submit hosts
- Run hosts

validatemasterhostname

- **Version 3.5 and earlier:** `validatemasterhostname` setting not available.
- **Version 4.0 and later:** `validatemasterhostname` setting available.

The `validatemasterhostname` setting enables the system administrator on the run host to perform an additional validation of the policy server host's name against the run host's name services. This action is done by looking up the connecting host's IP address and checking the run host's name services to see if it matches the `masterhost` variable that was looked up on the policy server host. When set to **yes**, the request is rejected if the names do not match.



Note: This keyword does not apply to `pbssh`. If it is present in the settings file, it does not have any effect on `pbssh` and is ignored.

**Example:**

```
validatemasterhostname yes
```

Default

```
validatemasterhostname no
```

Used on

Run hosts

validateclienthostname

- **Version 3.5 and earlier:** `validateclienthostname` not available.
- **Version 4.0 and later:** `validateclienthostname` setting available.

The `validateclienthostname` setting enables the system administrator on the policy server host to perform an additional validation of the submit host's name against the policy server host's name services. This action is done by looking up the connecting host's IP address and checking the policy server's name services to see if it matches the `clienthost` variable that was looked up on the submit host. When set to **yes**, the request is rejected if the names do not match.



Example:

```
validateclienthostname yes
```

Default

```
validateclienthostname no
```

Used on

Policy server hosts

Control Connections

addressfamily

- **Version 7.5 and earlier:** **addressfamily** setting not available.
- **Version 8.0 and later:** **addressfamily** setting available.

To support both IPv4 and IPv6 connections, Privilege Management for Unix and Linux uses protocol-independent methods for host name resolution. If Privilege Management for Unix and Linux is installed on a single stack node (ipv4-only or ipv6-only), the **addressfamily** setting may help make host name resolution more efficient by specifying which address family Privilege Management for Unix and Linux should use.

Syntax

```
addressfamily <ipv4 | ipv6 | any>
```

Valid Values

ipv4	Use IPv4 only.
ipv6	Use IPv6 only.
any	The network configuration on the host determines the address family. On dual- or hybrid-stack implementations, IPv4 or IPv6 may be requested/used. This is the default behavior if the keyword is not specified.



Example:

```
addressfamily ipv4
```

Default

No default value

Used On

- Policy server hosts
- Log hosts
- Submit hosts
- Run hosts

allowlocalmode

- **Version 4.0.0 and later:** `allowlocalmode` setting available.

Deprecated in favor of Optimized Run Mode.

When there is no need for ACA, or to record the finish of an event in the event logs or process keystrokes, local mode can bypass some of the overhead of using a full session. Local mode changes the way in which the Privilege Management for Unix and Linux job stream is set up. Using local mode bypasses `pblocald` entirely. The submit host asks `pbmasterd` to run its usual policy and log the start of the event. If accepted, the target program overlays the client instead of running through `pblocald`. In this case, there can be no logging of the exit status or forbidden or warning I/O sequences.

Local mode is usually started through the command line by using the `-l` option of `pbrun` (`pbrun -l command`), or by setting `runlocalmode` to `true` in the policy.

To disallow local mode, you can set `allowlocalmode` to `no` in the your settings file. In a submit host settings file, this setting disallows the use of the `-l` command line switch.

On a policy server host, setting `allowlocalmode` to `no` is the equivalent of:

```
runlocalmode = false;
readonly runlocalmode;
```



Note: This keyword does not apply to `pbssh`. If it is present in the settings file, it does not have any effect on `pbssh` and is ignored.



Example:

```
allowlocalmode no
```

Default

```
allowlocalmode yes
```

Used on

- Policy server hosts
- Run hosts
- Submit hosts



For more information, please see "[Optimized Run Mode Processing](#)" on page 21.

allowremotejobs

- **Version 3.5 and earlier:** `allowremotejobs` setting not available.
- **Version 4.0 and later:** `allowremotejobs` setting available.

Privilege Management for Unix and Linux allows commands to execute on machines other than the one that submits the request. This action can be specified, for example, through the `-h` option of `pbrun` or in the policy file. The `allowremotejobs` setting controls this feature.



Note: In version 7.1 and later, the `submitremotejobs` keyword also affects this feature. When the `submitremotejobs` keyword is not present, the `allowremotejobs` keyword functions exactly as prior versions functioned.

In version 7.0 and earlier, and with version 7.1 where the `submitremotejobs` keyword is not present on the submit host, setting `allowremotejobs` to `no` disables the use of the `-h` command line switch of `pbrun`.

On the run host, setting `allowremotejobs` to `no` makes `pblocald` reject all commands that did not originate on the run host.

On the policy server host, setting `allowremotejobs` to `no` makes `pbmasterd` reject all commands where the run host is different from the client host. In addition, `runhost` is set to the IP address of the submitting host and made read-only. Because the run host is resolved by IP address, this setting can be used to prevent run host spoofing in environments that do not use remote commands.



Note: This keyword does not apply to `pbssh`. If it is present in the settings file, it does not have any effect on `pbssh` and is ignored.



Example:

```
allowremotejobs no
```

Default

```
allowremotejobs yes
```

Used on

- Policy server hosts
- Run hosts
- Submit hosts

submitremotejobs

- **Version 7.0 and earlier:** `submitremotejobs` setting not available.
- **Version 7.1 and later:** `submitremotejobs` setting available.

Privilege Management for Unix and Linux allows commands to execute on machines other than the one that submits the request. This action can be specified, for example, through the **-h** option of **pbrun** or in the policy file.

On the submit host, setting **submitremotejobs** to **yes/no** enables/disables the use of the **-h** command line switch of **pbrun**. If the **submitremotejobs** keyword is not present, the **allowremotejobs** keyword is used to enable/disable this feature.

**Example:**

```
submitremotejobs no
```

Default

```
submitremotejobs yes
```

Used on

Submit hosts

tcpkeepalive

- **Version 4.0 and later:** **tcpkeepalive** setting available.

The **tcpkeepalive** setting enables TCP keepalive signals on all Privilege Management connections to or from the local host.

**Example:**

```
tcpkeepalive yes
```

Default

```
tcpkeepalive no
```

Used on

- Log hosts
- Policy server hosts
- Run hosts
- Submit hosts

Port Usage

Privilege Management for Unix and Linux uses sockets and ports for inter-program communication. These may be Unix/Linux domain socket ports or TCP/IP ports.

Privilege Management for Unix and Linux uses ports to form two types of connections:

- **Direct connection:** A request to a service on a well-known port such as:
 - **pbrun** to **pbmasterd**'s well-known port
 - **pbrun** to **pblogd**'s well-known port (local mode only)
 - **pbsh** or **pbksh** to **pbmasterd**'s well-known port
 - **pbsh** or **pbksh** to **pblogd**'s well-known port (local mode only)
 - **pbmasterd** to **pblogd**'s well-known port, if the configuration policy calls the **logmktemp()** function
 - **pbmasterd** to **pblocald**'s well-known port
 - **pblocald** to **pblogd**'s well-known port
 - A browser to **pbguid**'s well-known port
 - **pbguid** to **pbmasterd**'s well-known port
 - REST services communicating between hosts

For TCP/IP, the **allownonreservedconnections** setting can affect this type of connection.

- **Dynamic connection:** This connection type is an optimization which enables two programs that are both connected to a common third program to connect directly to each other when the common program is no longer needed.

For example, **pbrun** starts **pbmasterd**, which in turn starts **pblocald**. When the **pbmasterd** work is done, it can instruct **pbrun** and **pblocald** to connect directly to each other. This action streamlines the network traffic and enables **pbmasterd** to exit without affecting the other two programs, thus saving resources.

Two dynamic connections that Privilege Management for Unix and Linux uses are:

- **pbrun** and **pblocald** when **pbmasterd** is finished. The **pbrunreconnection** variable can control the direction of the connection.
- **pblocald** and **pblogd** when **pbmasterd** is finished and the configuration policy uses the **logmktemp()** function. The direction of the connection can be controlled by the **pblogdreconnection** variable in the policy language.

For TCP/IP ports in each case, one program opens a listening port (in the range **minlisteningport** to **maxlisteningport**) then waits for the other to connect to it from an outgoing port (in the range **minoutgoingport** to **maxoutgoingport**).

For Unix/Linux domain sockets, temporary sockets are formed in the **localsocketdir** directory.



For more information, please see "[allownonreservedconnections](#)" on page 110

TCP/IP Ports

TCP/IP uses numeric ports from **1** and **65535**. Reserved ports, which are accessible only to **root**, are in the range **1** to **1023**. Non-reserved are in the range **1024** to **65535**.

allownonreservedconnections

- **Version 4.0.0 and later:** `allownonreservedconnections` setting available.

Privilege Management programs can initiate connections using reserved ports (for example, ports 600-1023), as well as non-reserved ports. Support for non-reserved ports was introduced in v3.0.5. The default behavior is to check for connections on reserved ports. This check can be disabled by setting `allownonreservedconnections` to **yes**.



Example:

```
allownonreservedconnections yes
```

Default

```
allownonreservedconnections yes
```

Used on

- Log hosts
- Policy server hosts
- Run hosts

minlisteningport and maxlisteningport

- **Version 4.0.0 and later:** `minlisteningport` and `maxlisteningport` settings available.

When a dynamic connection is needed between two Privilege Management for Unix and Linux programs, one program opens a listening port in the range between `minlisteningport` and `maxlisteningport`.

To control the range of ports that are used to listen for these dynamic connections, you can set `minlisteningport` and `maxlisteningport` to define the range of allowable ports.



Example:

```
minlisteningport 10000  
maxlisteningport 10200
```

Default

```
minlisteningport 1024  
maxlisteningport 65535
```

Used on

- Log hosts
- Policy server hosts
- Run hosts
- Submit hosts

minoutgoingport and maxoutgoingport

- **Version 4.0.0 and later:** `minoutgoingport` and `maxoutgoingport` settings available.

When a Privilege Management program needs to contact another program, the program opens an outgoing port in the range between **minoutgoingport** and **maxoutgoingport**. This range is used for connections to a well-known service port and for dynamic connection.

If you want to use unreserved ports, then make sure that **allownonreservedconnections** is set to **yes** for the host that receives the connection.



Example:

```
minoutgoingport 20000
maxoutgoingport 20200
```

Default

```
minoutgoingport 600
maxoutgoingport 1023
```

Used on

- Log hosts
- Policy server hosts
- Run hosts
- Submit hosts



For more information, please see "[allownonreservedconnections](#)" on page 110.

Program Ports

- localport
- logport
- masterport

- pbrestport
- guiport
- sguiport
- Solrport
- syncport
- rcswebsvcport

i For more information, please see the following:

- "[localport](#)" on page 93
- "[logport](#)" on page 100
- "[masterport](#)" on page 83
- "[pbrestport](#)" on page 272
- "[guiport](#)" on page 163
- "[sguiport](#)" on page 164
- "[Solrport](#)" on page 153
- "[syncport](#)" on page 171
- "[rcswebsvcport](#)" on page 150

Unix/Linux Domain Sockets

Unix/Linux Domain Sockets are temporary files used for dynamic connections created in the **localsocketdir** directory.

localsocketdir

- **Version 3.5 and earlier:** **localsocketdir** setting not available.
- **Version 4.0 and later:** **localsocketdir** setting available.

The **localsocketdir** setting provides the path to a directory for Unix/Linux domain socket connections used in dynamic connections. The directory should be owned by root and should be readable and writable only by root (for example, **drwx-----**).

o **Example:**

```
localsocketdir /var/pbLocalSockets
```

Default

No default value

Used on

- Log hosts
- Policy server hosts
- Run hosts
- Submit hosts

Auditing and Logging

Privilege Management for Unix and Linux provides event logs and audit logs for:

- Authorization Event Logging
- Audit Events
- Change Management Events
- License Events
- File Integrity Monitor Events
- Advanced Keystroke Action Events
- Session Logging
- I/O Logging
- I/O Log Indexing and Searching
- AD Bridge Event Logging
- BeyondInsight Event and IO Logging
- Log Synchronization
- Diagnostic Logging

Authorization Event Logging

Privilege Management for Unix and Linux records the following authorization events in the event log file on the log host or Policy Server host (if not using a log server):

- Accept and Reject events (including policy variables) for all tasks.
- Keystroke patterns set by **keystrokeactions** and task finish information for jobs that are not run in local mode. This file can be reviewed through the Privilege Management for Unix and Linux GUI or with the **pblog** command.

The event log is stored in a flat file whose path may be defined by the **eventlog** variable in the policy, **eventdestinations** setting, or **eventlog** setting.

Store Eventlog Records in a Database

SQLite Database


Starting with v10.3.0, event log records can be written to an SQLite database using the setting below which can optionally accept a path name:

```
eventdestinations authevt=db[=</path/to/database>]
```

If no specific file is given, the path in the eventlog variable is used (it suffixes `.db` to the filename, if the filename does not already include a `.db` at the end).

This is the default eventlog type for Privilege Management for Unix and Linux v10.3.0, v10.3.1, and v10.3.2.

The `pblog` program will automatically detect if an eventlog is an SQLite database and read it accordingly. To force the SQLite database read mode, use the `pblog` option `--db`.


 For more information, please see ["eventdestinations" on page 116](#).

MySQL and Oracle Database using ODBC Connectors

Starting with v10.3.0, event log records can be sent to an Oracle or MySQL database using the ODBC connector on the log server.

If **eventdestinations** is set to **authevt=odbc=<MyDSN>**, the event log records are logged to the ODBC DSN specified in the setting and found in the ODBC configuration files. The value of **eventlog** in **pb.settings** is ignored, and the event log records are written to the ODBC DSN specified in **odbc.ini** and **odbcinst.ini** in the **odbcinidir** directory (or default **/opt/pbul/etc**). If the **eventlog** variable is defined in the policy, then the event log records are written to both the path and file in the policy (in flat file format), and in the ODBC database. For **pblog**, use the option **--odbc -f <MyDSN>** to read the MySQL/Oracle database.

Event destinations can be combined, using commas, to enable logging to multiple services.

 **Example:**

```
eventdestinations authevt=db,odbc=MySQL,/var/log/eventlog.flatfile
```

The setting **anydestinationsufficient** controls how the events are written to multiple destinations:

- If set to **no** (default), it requires that the event be written to all defined destinations (db, odbc, flat-file) before it is removed from the message queue. If one destination is down, it keeps the event on the message queue until the event is successfully written to all destinations.



Note: If the event was already written to a destination (in case of a flat file), the next attempt may create a duplicate record.

- If it is set to **yes**, it attempts to write each event to every destination. If any one of them fails, as long as the attempt to any destination succeeds, it proceeds to the next event.

In a BeyondInsight integrated environment, Privilege Management for Unix and Linux can optionally send the events to BeyondInsight Web Services. Events can then be viewed on BeyondInsight GUI and can also be queried using the standard BeyondInsight report plugin.



For more information, please see the following:

- For instructions on installing the ODBC connectors on the log servers, the [Installation Guide](https://www.beyondtrust.com/docs/privilege-management/unix-linux/index.htm) at <https://www.beyondtrust.com/docs/privilege-management/unix-linux/index.htm>
- "BeyondInsight Event Logging" on page 150

eventdestinations

- **Version 9.4.1 and earlier:** **eventdestinations** setting not available.
- **Version 9.4.3 and later:** **eventdestinations** setting for Audit events available.
- **Version 10.3.0 and later:** **eventdestinations** setting for Authorization events available.

The **eventdestinations** setting allows the configuration of where events are logged. Prior to v10.3.0, this setting is used only for audit events. Starting in v10.3.0, it has been augmented to support authorization event logging using the keyword **authevt**.




For more information about configuring **eventdestinations** setting for audit events, please see "Audit Events" on page 123.


Syntax

```
eventdestinations authevt=<destination>
```

Where destination can be one or a combination of database, syslog, flat file, or passed into a script or binary for processing:

- **db:** Write the event log records to an SQLite database whose path is specified in the **eventlog** setting.
- **db=/path/to/database.db:** Write the event log records to the named SQLite database.
- **/path/to/flatfile:** Write the event log records to flat file.
- **odbc=DSN:** Log the event records to the configured ODBC DSN (Data Source Name).
- **syslog:** Write the event log records to the local syslog service, using the syslog configuration in **pb.settings**.
- **/path/to/script:** Will log the event to the script or binary specified.

 **Note:** If `eventdestinations` is omitted or commented out in the settings, Privilege Management for Unix and Linux defaults to using flat file eventlog format.

 **Note:** Within the `authvt` group, a combination of each destination type can be specified, separated by commas. If combining with Audit events options, each new group should be delimited with a space.

authvt Usage

<code>authvt=db</code>	<p>The path to the default database is set using either <code>eventlog</code> setting, or it defaults to <code>{var usr}/{adm log}/pb.eventlog.db</code>.</p> <p>If the <code>eventlog</code> variable is defined in the policy, then the event log records are written to both the path and file in the policy (in flat file format), and the path/file defined by the <code>eventlog</code> setting (in SQLite database format).</p> <p>If <code>eventlog</code> in <code>pb.settings</code> is set to a filename without <code>.db</code> at the end, a <code>.db</code> is implicitly added. For example, if <code>eventlog</code> is set to <code>/var/log/pb_event_file</code>, the file name is changed to <code>/var/log/pb_event_file.db</code>.</p>
<code>authvt=db=/path/to/database.db</code>	<p>This option allows different database names in the <code>eventdestinations</code> setting.</p> <p>The value of <code>eventlog</code> in <code>pb.settings</code> is ignored. When writing the event log records the path/file is <code>/path/to/database.db</code>. If the <code>eventlog</code> variable is defined in the policy, then the event log records are written to both the path/file defined in the policy (in flat file format), and the path/file defined by this setting in SQLite database format.</p>
<code>authvt=/path/to/flatfile</code>	<p>This option logs events in the flat-file format specified in the setting.</p> <p>The value of <code>eventlog</code> in <code>pb.settings</code> is ignored, and the file name (<code>/path/to/flatfile</code>) is used when writing the event log records.</p> <p>If the <code>eventlog</code> variable is defined in the policy, then the flat-file event log is created using that path instead of the value (<code>/path/to/flatfile</code>) specified in this setting.</p>
<code>authvt=odbc=MyDSN</code>	<p>This option logs events to the ODBC DSN specified in the setting and found in the ODBC configuration files.</p> <p>The value of <code>eventlog</code> in <code>pb.settings</code> is ignored, and the event log records are written to the ODBC DSN set in <code>odbc.ini</code> and <code>odbcinst.ini</code> in the directory <code>odbcinidir</code> (or default <code>/opt/pbul/etc</code>).</p> <p>If the <code>eventlog</code> variable is defined in the policy, then the event log records are written to both the path/file defined in the policy (in the flat file format), and in the ODBC database.</p>
<code>authvt=syslog</code>	<p>This option logs events to the local hosts syslog service in a JSON format.</p>
<code>authvt=/path/to/script</code>	<p>This option logs events to a script or binary in JSON format.</p>

To avoid running out of resources during heavy load while support authentication events to a program (`eventdestinations authvt=|program`), `pblhttpd-svc` must be started with unlimited `NOFILES` and `NPROC`, at least.

i For more information, please see "[pblighttpd Service](#)" on page 515.

o **Example:**

```
eventdestinations authevt=db
```

o **Example:** Event destinations can be combined, separated by commas, to enable logging to multiple services:

```
eventdestinations authevt=db, /var/adm/pb.eventlog.flat, odbc=MySQL, syslog
```

o **Example:** Authorization Event and Audit Event destinations can be combined using a space delimiter:

```
eventdestinations authevt=db chgmt=db, syslog fimrpt=|/mydir/process license=db, syslog
```

Default

```
eventdestinations authevt=<eventlog>
```

Used On

- Log servers
- Policy server hosts (If a log host is not used)

i For more information on how to configure for other types of supported events, please see "[eventdestinations](#)" on page 123.

eventlog

- **Version 4.0.0 and later:** **eventlog** setting available.
- **Version 10.3.0 and later:** **eventlog** setting usage modified.

The **eventlog** setting specifies the default location of the flat file or SQLite database event log.

If **eventdestinations** setting is omitted from the settings file or does not have an explicit "authevt=" designation, Privilege Management for Unix and Linux writes authorization events in flat file format to the location in the eventlog setting by default.

The **eventdestinations** setting "**authevt=db**" configures authorization events to be written to an event log in SQLite database format, in which case, the **eventlog** setting provides the pathname of the event log (with **.db** suffix).

Any parent directory in the path is automatically created.

When **pblog** is invoked with no specific eventlog filename and **eventdestinations** setting does not explicitly define a path for a flat file (authvt=<path>) or SQLite database (authvt=db=<path>), it will try to read the path in the **eventlog** setting. If **eventdestinations** is set to "authvt=db", pblog will implicitly append ".db" to the path first before reading it.

**Example:**

```
eventlog /var/log/my.event.log
```

Default

Depending on the operating system standards, this can be any of the following:

```
eventlog /var/log/<prefix>pb.eventlog<suffix>
```

```
eventlog /var/adm/<prefix>pb.eventlog<suffix>
```

```
eventlog /usr/adm/<prefix>pb.eventlog<suffix>
```

Used on

- Log hosts
- Policy server hosts (If a log host is not used)

syslog

- **Version 4.0.0 and later:** **syslog** setting available.

Enables sending diagnostic messages to syslog, using the **facility** setting.

To enable syslog recording of diagnostic messages, set **syslog** to **yes**.

**Example:**

```
syslog yes
```

Default

```
syslog yes
```

Used on

- Log hosts
- Policy server hosts
- Run hosts
- Submit hosts

 For more information, please see "facility" on page 120.

facility

- **Version 4.0.0 and later:** **facility** setting available.

The messages that the Privilege Management for Unix and Linux programs transmit to the syslog facility are labeled with a syslog level. This level (as well as the severity that is specified internally to Privilege Management for Unix and Linux on a per message basis) is used by the syslog facility and handled according to the rules in the syslog configuration file (typically **/etc/syslog.conf**). If Privilege Management for Unix and Linux messages are not appearing in syslog, then verify that **syslog.conf** is handling the facility that you are using as expected.

To specify the syslog facility for logging to the Unix/Linux syslog subsystem, set **facility** appropriately. Some commonly available facilities include:

- **LOG_AUTH**
- **LOG_AUTHPRIV** (Linux).
 - Only supported in Privilege Management for Unix and Linux 7.1.0 and later.
- **LOG_USER**
- **LOG_MAIL**
- **LOG_DAEMON**
- **LOG_LPR**
- **LOG_NEWS**
- **LOG_UUCP**
- **LOG_CRON**
- **LOG_LOCAL0** through **LOG_LOCAL7**



Example:

```
facility LOG_DAEMON
```

Default

```
facility LOG_AUTH
```


Used on

- Log hosts
- Policy server hosts if a log host is not used

anydestinationsufficient

- **Version 10.2.0 and earlier:** `anydestinationsufficient` setting not available.
- **Version 10.3.0 and later:** `anydestinationsufficient` setting available.

This setting controls how the events are written to multiple destinations.

When set to **no**, `anydestinationsufficient` writes an event to all defined destinations (db, odbc, flat file). All destinations must be up and running. If one destination is down, the event is retained in the message router write queue until it is successfully written to all event log destinations. This is the default behavior.

When set to **yes**, `anydestinationsufficient` writes the event to one destination at a time. If one destination is down, and the event has already been written to at least one database (db, odbc, flat file), then it skips the destination for that event, and moves on to the next event. The database that is down does not have all the events. However, `pblog` can be used to get the events from one destination and write it to another.



Example:

```
anydestinationsufficient yes
```

Default

no

Used On

Log servers

odbcinidir

- **Version 10.2.0 and earlier:** `odbcinidir` setting not available.
- **Version 10.3.0 and later:** `odbcinidir` setting available.

The `odbcinidir` setting provides the path where `odbc.ini` and `odbcinst.ini` are read from when `eventdestinations` is set to `authvt=odbc=<DSN>`.

- **odbc.ini:** Provides authentication details for the ODBC database.
- **odbcinst.ini:** Includes driver details where shared libraries reside.

**Example:**

```
odbcinidir /etc/odbc
```

Default

```
odbcinidir /opt/<prefix>pbul<suffix>/etc
```

Audit Events

There are various audit events that can be enabled and stored within Privilege Management for Unix and Linux . These include Configuration Change Management, File Integrity Monitoring events, Advanced Keystroke Action events, and License events. These events are enabled on clients and servers, and are logged on the log server. Each type of event has a taxonomy to distinguish the type of data that it contains. There are various settings that affect the audit event log.

eventdb

- **Version 8.5.0 and earlier:** **eventdb** setting not available.
- **Version 9.0.0 and later:** **eventdb** setting available.

The **eventdb** setting details where Audit events are stored on the log server if there is no specific configuration using the **eventdestinations** setting. If a relative path is specified, the **basedir** setting is used to derive the full path.



Example:

```
eventdb /mypath/pbevent.db
```

Default

```
eventdb /opt/<prefix>pbul<suffix>/dbs/pbevent.db
```

Used On

All hosts

eventdestinations

- **Version 9.4.1 and earlier:** **eventdestinations** setting not available.
- **Version 9.4.3 and later:** **eventdestinations** setting for Audit events available.
- **Version 10.3.0 and later:** **eventdestinations** setting for Authorization events available.

The **eventdestinations** setting allows the configuration of where each taxonomy of Audit events is logged. Starting in v10.3.0, this setting can also be used to configure destinations for authorization events.

Syntax

```
eventdestinations <taxonomy>=<destination> ...
```

Taxonomy	Event Type
chgmt	Configuration Change Management
client	Client Registration
fimrpt	File Integrity Monitoring
errlog	Miscellaneous Error Logging via REST (including ACA and user-defined errors)
aka	Advanced Keystroke Action
license	License Events
policydbg	Policy Language Debugging
authvt	accept, reject, finish, keystroke events

The destination can be one or more database, syslog, flat text file, or passed into a script or binary for processing:

- **db=/path:** Outputs the event to a database.
- **db:** Outputs the event to the database specified in the **eventdb** settings
- **syslog:** Outputs the event to the local syslog service, using the syslog configuration in **pb.settings**.
- **/directory/file:** An absolute path to a file which is appended with the event in text in the specified format.
- **!/directory/script:** Passes the event on standard input into the script or binary specified.



Note: Within each taxonomy, one or more destinations can be specified, separated by commas. If specifying more than one taxonomy or combining with Authorization Event Logging **eventdestinations** option, each group should be delimited with a space.



Example:

```
eventdestinations chgmt=db
```

Event destinations can be combined, separated by commas, to enable logging to multiple services:

```
eventdestinations chgmt=db,syslog,/var/adm/pbchgmt.log
```

Multiple Audit Event and Authorization Event destinations can be combined using a space delimiter:

```
eventdestinations chgmt=db,syslog fimrpt=!/mydir/process license=db,syslog
```

```
eventdestinations authvt=db chgmt=db,syslog fimrpt=!/mydir/process license=db,syslog
```

Default

By default, all events are logged to the database specified by **eventdb**.

Used On

Log servers

 For more information, please see **eventdestinations** setting description in "Authorization Event Logging" on page 115.

eventformats

- **Version 9.3.3 and earlier:** **eventformats** setting not available.
- **Version 9.4.4 and later:** **eventformats** setting available.

Events that are not logged into a database can be logged in two different formats:

- Labeled Comma Separated Values, where values take the form **<attribute_name>=<value>,...**
- JSON format

Example:

```
eventformatschgmgmt=json license=csv
```

Default

By default, all events are logged in JSON format.

Used On

Log servers

Message Router

With the introduction of Privilege Management for Unix and Linux v10.1.0, a Fast Message Router was developed to better cope with high volumes of event and log information. The log server communicates all of its updates to the Message Router, which then logs the audit and log information to the appropriate places, including the event log, the BeyondInsight Event queues, and the SOLR /lologcloseaction queues.

The **pblighttpd** service, which previously started the REST and Scheduler services, now also starts the Message Router services. If the Message Router is down, the log server stores all of its data in a temporary queue until the Message Router service is available again.

messengeroutersocketpath

- **Version 10.0.1 and earlier:** **messengeroutersocketpath** setting not available.
- **Version 10.1.0 and later:** **messengeroutersocketpath** setting available.

This setting defines the absolute path to the Message Router work area, where sockets and temporary files are stored while the Message Router is unavailable.



Example:

```
messengeroutersocketpath /opt/pb/pb_msgrouter
```

Default

```
messengeroutersocketpath /opt/<prefix>pbul<suffix>/msgrouter
```

Used On

- Policy servers
- Log servers

messengerouterqueuesize

- **Version 10.0.1 and earlier:** **messengerouterqueuesize** setting not available.
- **Version 10.1.0 and later:** **messengerouterqueuesize** setting available.
- **Version 22.1 and later:** **messengerouterqueuesize** setting syntax updated.

This setting specifies the number of temporary queue entries (and beginning in 22.1.0, also specifies the entry size) for the Message Router (used to route events to event logs, BIUL, Solr, SIEM, and other databases). Generally the default is acceptable, however, in large installations with many events logged per second, it might need to be increased.

Syntax in Versions 10.1.0—21.1

```
messengerouterqueuesize x
```

The maximum value is 4095.



Example:

```
MessageRouterqueuesize 1000
```

Default

```
MessageRouterqueuesize 800
```

Syntax in 22.1.0 and Later

In 22.1.0, the previous syntax is still supported, however, additional syntax allows for much finer control:

```
messengerouterqueuesize ( x | identifier=numchunks,[chunksize])  
[identifier=numchunks[,chunksize]]*
```

The **x** value is compatible with the prior version syntax:

```
messengerouterqueuesize 800
```

The **x** value defines a default number of queue entries for any service not further defined in the value.

The valid identifiers are:

- **eventlog**
- **ioc**: The I/O log close action.
- **iol**: The I/O log cache service.
- **intprod**: The integrated Products Message Router service (sends events to BIUL).
- **siem**: The Message Router SIEM service (sends events to Elasticsearch).

Each **numchunks** value has a minimum of 50 and a maximum of 400000. The **chunksize** values are in **K** units.

For example, "... eventlog=200,8 ..." allows for 200 8K chunks for events destined for an eventlog. This might be used by 200 events sized up to 8K each, or 100 events sized between 8K and 16K, or 50 events sized 32K – or any combination of events of different sizes.

**Example:**

```
MessageRouterQueueSize eventlog=22000,8 intprod=22000,8 siem=40000,8 ioc=200,16  
iol=200,16
```

Default

```
MessageRouterQueueSize eventlog=200,8 intprod=200,8 siem=200,8 ioc=200,16 iol=200,16
```

Used On

- Policy servers
- Log servers

messagingrouterclosewait

- **Version 10.0.1 and earlier:** `messagingrouterclosewait` setting not available.
- **Version 10.1.0 and later:** `messagingrouterclosewait` setting available.

This setting specifies the maximum time a policy or log server waits, in seconds, for the Messaging Router to become available. If it cannot contact the Messaging Router after this duration, it writes the entries to temporary queues for later processing.

**Example:**

```
messagingrouterclosewait 120
```

Default

```
messagingrouterclosewait 30
```

Used On

- Policy servers
- Log servers

writetimetimeouts

- **Version 10.0.1 and earlier:** `writetimetimeouts` setting not available.
- **Version 10.1.0 and later:** `writetimetimeouts` setting available.

This setting specifies the timeouts for various actions for write queue operation. Although we do not recommend that these are changed in most instances, there may be scenarios, usually when hosts experience high load levels, that configuration of these timeouts can improve performance. The timeout operations include:

- openread
- openwrite
- write
- lock

Each timeout has three elements:

- The maximum timeout in milliseconds
- The incremental delay each time the operation is blocked
- A backoff component that can be used to increase the delay time each time the operation blocks

**Example:**

```
writequeuetimeouts openread=1000,20,1.2 openwrite=5000,20,1.2 write=2000,10,1.0  
lock=30000,50,2.0
```

Default

No default value

Used On

- Policy servers
- Log servers

writequeuepath

- **Version 10.0.1 and earlier:** `writequeuepath` setting not available.
- **Version 10.1.0 and later:** `writequeuepath` setting available.

The `writequeuepath` is the absolute path to a directory that will hold all the temporary write queues. This path should have enough disk space to cope with high volumes of data should the message router become unavailable.

**Example:**

```
writequeuepath /opt/pbul/msgqueue
```

Default

```
writequeuepath /opt/<prefix>pbul<suffix>/msgrouter
```

Used On

- Policy servers
- Log servers

writequeueenum

- **Version 10.0.1 and earlier:** `writequeueenum` setting not available.
- **Version 10.1.0 and later:** `writequeueenum` setting available.

The `writequeueenum` setting allows the configuration of the maximum number of write queues that can be stored in the event that the Message Router becomes unavailable. The minimum value of `writequeueenum` is 10 and the maximum is 9999.



Note: This setting should only be increased in the event of contention due to very high load.



Example:

```
writequeueenum 9999
```

Default

```
writequeueenum 999
```

Used On

- Policy servers
- Log servers

Change Management Events

Change Management events are configured on the client by enabling **changemanagementevents** in the **pb.settings**, and on the primary log server by specifying the **eventdb** setting. This logs all changes made to the Configuration and Settings and to the Role Based Policy databases. When the setting is enabled, all changes require a message, which is logged alongside the **username**, **date/time** and the details of the actual change. The events are sent to the log server defined in the **pb.settings** and can be retrieved via REST or locally on the log server with the **pbdbutil --evt** option.



Example: The administrator changes the **pb.settings** file and reimports it:

```
# pbdbutil --cfg -m "Change to pb.settings" -i /etc/pb.settings
{"fname":"/etc/pb.settings","version":4}
```

Then the administrator adds a new Role into the Role Based Policy database:

```
# pbdbutil --rbp -m "New role" -u '{ "role" : {"id":2,"name":"new
role","rorder":2,"description":"new role for
admin","disabled":0,"risk":0,"action":"A","iolog":null,"script":null}}'
```

The Change Management Events can be accessed on the primary log server:

```
# pbdbutil --evt -s chgmt
{
  "hostname": "pbuild", "evtname": "file_import", "service": "pbdbutil9.0.0-
14", "who": "admin1", "severity": 16, "utc": "2015-05-21 11:53:07",
  "progname": "pbdbutil9.0.0-14 ", "version": "9.0.0-14 ", "arch": "x86_
64_linuxA",
  "data": {
    "fname": "/etc/pb.settings", "version": 4, "msg": "Change to pb.settings",
    "sid": 33235, "pid": 34378, "uid": 0
  }
}
{
  "hostname": "pbuild", "evtname": "put", "service": "pbdbutil9.0.0-14",
  "who": "admin1", "severity": 16, "utc": "2015-05-21 11:56:35", "progname":
  "pbdbutil9.0.0-14 ", "version": "9.0.0-14_debug", "arch": "x86_64_
linuxA",
  "data": {
    "id": 2, "description": "new role for admin", "risk": 0, "action": "A",
    "name": "new role", "rorder": 2, "disabled": 0, "iolog": null, "script":
    null, "sid": 33235, "pid": 34423, "uid": 0
  }
}
```



For more information, please see ["pbdbutil, pbadmin" on page 378](#).

License Events

The licensing system introduced in version 10.0.1 allows the logging of all license history events. This produces an event for every license check, and as such can impose an increased load on all server components. When it is enabled, each policy server, File Integrity Server, Advanced Keystroke Action Server, etc, sends an event to the log server to be logged as specified in **eventdestinations**.

licensehistory

This setting enables the production of license history events on all servers. The setting is synchronized from the primary license server across the whole Privilege Management for Unix and Linux installation.



Example:

```
licensehistory yes
```

Default

```
licensehistory no
```

Used On

All servers

File Integrity Monitor Events

The File Integrity Monitor component can be configured to send summary events from reports as an Audit event. The **eventdestinations** setting details how these events are processed, and can be used to monitor the status of the service. This setting should be specified on all the File Integrity Monitor Policy Servers.

fileintegrityevents

- **Version 9.3.0 and earlier:** **fileintegrityevents** setting not available.
- **Version 9.4.0 and later:** **fileintegrityevents** setting available.

This setting enables the production of File Integrity Report events, which are logged in the log server.



Example:

```
fileintegrityevents yes
```

Default

```
fileintegrityevents no
```

Used On

All File Integrity Servers

Advanced Keystroke Action

The Advanced Keystroke Action component can be configured to send events for every successful session, and command run on a remote server or network appliance.

advkeystrokeactionevents

- **Version 9.3.3 and earlier:** `advkeystrokeactionevents` setting not available.
- **Version 9.4.4 and later:** `advkeystrokeactionevents` setting available.

The `advkeystrokeactionevents` setting enables the production of Advanced Keystroke Action events, which are logged on the log server.



Example:

```
advkeystrokeactionevents yes
```

Default

```
advkeystrokeactionevents no
```

Used On

All Advanced Keystroke Action servers

advkeystrokeactionpolicydb

- **Version 9.4.1 and earlier:** `advkeystrokeactionpolicydb` setting not available.
- **Version 9.4.3 and later:** `advkeystrokeactionpolicydb` setting available.

The `advkeystrokeactionpolicydb` setting defines the path to the Advanced Keystroke Action policy database which stores all the AKA policies.

This file is created in `basedir` by default, unless the file name starts with `/`.



Example:

```
advkeystrokeactionpolicydb /etc/pbakapolicy.db
```

Default

```
advkeystrokeactionpolicydb /opt/<prefix>pbul<suffix>/dbs/pbadvkeystrokeactionpolicy.db
```

Used On

Advanced Keystroke Action Policy servers

 For more information, please see "[databasedir](#)" on page 48

advkeystrokeactioncachedb

- **Version 9.4.1 and earlier:** `advkeystrokeactioncachedb` setting not available.
- **Version 9.4.3 and later:** `advkeystrokeactioncachedb` setting available.

The `advkeystrokeactioncachedb` setting defines the path to the Advanced Keystroke Action profile cache database that is kept on the client so that it always has access to the latest AKA profiles.

Example:

```
advkeystrokeactioncachedb /etc/pbakacache.db
```

Default

```
advkeystrokeactioncachedb /opt/<prefix>pbul<suffix>/dbs/pbadvkeystrokeactioncache.db
```

Used On

Advanced Keystroke Action clients

advkeystrokeactionlog

- **Version 9.4.1 and earlier:** `advkeystrokeactionlog` setting not available.
- **Version 9.4.3 and later:** `advkeystrokeactionlog` setting available.

The `advkeystrokeactionlog` contains the path name of the Advanced Keystroke Action diagnostic log file.

Example:

```
advkeystrokeactionlog /var/log/pbakapolicy.log
```

Default

No default value

Used on

Advanced Keystroke Action Policy servers

Session Logging

Privilege Management for Unix and Linux records the start of all commands and the finish of all commands not run in local mode. The session start and finish events can also be logged by using the following:

- System **wtmp** or **wtmpx** files (**recordunixptysessions**)
- Syslog system (**syslogsessions**)
- PAM system (**pamsessionservice**)

recordunixptysessions

- **Version 3.5 and earlier:** **recordunixptysessions** setting not available.
- **Version 4.0 and later:** **recordunixptysessions** setting available.

The **recordunixptysessions** setting controls whether command start and finish events are logged to the run host **utmp** or **utmpx** files. When set to **yes**, the events are logged.



Note: This keyword does not apply to **pbssh**. If it is present in the settings file, it does not have any effect on **pbssh** and is ignored.



Example:

```
recordunixptysessions no
```

Default

```
recordunixptysessions yes
```

Used on

Run hosts



Note: If you are using **pamsessionsservice**, then you might need to set **recordunixptysystems** to **no** to avoid duplicate entries in your **utmp** or **utmpx** files.



Note: When the login shell is a Privilege Management shell and I/O logging is on, an additional **pty** is created, which is logged in the run host's **utmp** log. Note that the **ut_host** field is set to the run host value, not the remote host, because this **pty** originated on the run host.

syslogsessions

- **Version 4.0.0 and later:** `syslogsessions` setting available.

The `syslogsessions` setting controls whether command start and finish events are logged to the run host syslog system. When set to **yes**, the events are logged.



Note: This keyword does not apply to `pbssh`. If it is present in the settings file, it does not have any effect on `pbssh` and is ignored.



Example:

```
syslogsessions yes
```

Default

```
syslogsessions no
```

Used on

Run hosts



Note: If you are using `pamsessionsservice`, then you might need to set `syslogsessions` to `no` to avoid duplicate syslog entries.

I/O Logging


Privilege Management can maintain I/O logs of sessions under control of the configuration policy language. The **passwordlogging** and **rootshelldefaultiolog** settings provide additional system-wide control of the I/O logs. On policy server hosts and log hosts, you can control how much file system space is used with the **logreservedfilesystems** and **logreservedblocks** settings.

Privilege Management records I/O logs in log files on the log host or policy server host (if not using a log server).

I/O Log Indexing and Searching

In a BeyondInsight integrated environment, using Solr servers, each log server and policy server host can communicate with a Solr server, submitting Privilege Management for Unix and Linux I/O log output data for indexing.

BeyondInsight provides a search GUI, allowing users to search indexed I/O logs.

 For more information about enabling this feature, please see "[BeyondInsight I/O Log Indexing and Searching](#)" on page 153.


iologack

- **Version 6.0 and earlier:** **iologack** setting not available.
- **Version 6.2.5 and later:** **iologack** setting available.

The **iologack** setting enables a log host to send an **acknowledgement** to the submit host after the log host writes each I/O log data segment. Using this setting can prevent data integrity problems and prevent the submit host from hanging when there are network interruptions or if the log host becomes unavailable during an I/O logging session. However, enabling **acknowledgements** can increase network traffic and degrade system performance.

The submit host waits for **acknowledgement** for a period of time that is determined by the **logserverprotocoltimeout** setting. If **logserverprotocoltimeout** is set to a value other than **-1**, then the timeout period is 10 seconds. Otherwise, the timeout period is the value of **logserverprotocoltimeout**.

For **acknowledgements** to be sent, the **iologack** setting on the submit host and the log host must both be set to **yes**.

 **Example:**

```
iologack yes
```

Default

```
iologack no
```

Used on

- Submit hosts
- Log hosts

passwordlogging

- **Version 4.0.0 and later:** **passwordlogging** setting available.

It might be desirable to control whether passwords can be logged to a greater extent than using the variable **lognpassword** alone. Setting **passwordlogging** to **never** suppresses all text portions of the input stream that are not echoed in the output stream. This action also sets the configuration policy language variable **lognpassword** to **never** and makes it read-only.

Valid Values

- **allow**
- **never**



Example:

```
passwordlogging allow
```

Default

```
passwordlogging never
```

Used on

- Policy server hosts
- Run hosts
- Submit hosts



For more information about the **lognpassword** variable, please see the [Privilege Management for Unix and Linux Policy Language Guide](https://www.beyondtrust.com/docs/privilege-management/unix-linux/index.htm) at <https://www.beyondtrust.com/docs/privilege-management/unix-linux/index.htm>.

rootshelldefaultiolog

- **Version 4.0.0 and later:** **rootshelldefaultiolog** setting available.

When root runs a Privilege Management for Unix and Linux shell (for example, **pbsh** or **pbksh**), and a policy server daemon cannot be reached, that shell records an I/O log for the root session. Because no policy server can be reached, **rootshelldefaultiolog** provides a default emergency I/O log. If the file name is not unique, then Privilege Management for Unix and Linux adds a unique 6-character suffix to the name.



Example:

```
rootshelldefaultiolog /var/logs/root.default.iolog
```

Default

```
rootshelldefaultiolog /pbshell.iolog
```

Used on

Submit hosts by **pbksh** and **pbsh** when a policy server host is not available.

logreservedfilesystems and logreservedblocks

- **Version 4.0.0 and later:** **logreservedfilesystems** and **logreservedblocks** settings available.

The **logreservedfilesystems** and **logreservedblocks** settings enable the administrator to control free space on the **logreservedfilesystems** file systems, and cause an immediate failover if the log host's free space falls below **logreservedblocks**.

If the number of free 1KB blocks falls below **logreservedblocks** on any of the file systems that are specified in any of the **logreservedfilesystems** on the log host, then the log daemon immediately refuses any new requests, causing an immediate failover. The same happens on the policy server host if you are not using a log server.

If the free space in any of the file systems containing **/var/log** or **/usr/log** falls below 10,000 blocks, then new requests are rejected. Requests that are already in progress are allowed to continue.



Example:

```
logreservedfilesystems /var /usr/log  
logreservedblocks 2000
```

Default

```
logreservedblocks 0
```

No default value for **logreservedfilesystems**.

Used on

- Log hosts
- Policy server hosts if a log host is not used

Customized Syslog Formatting

For syslog logging, you can specify the format and select specific fields to be written to the syslog file for accept, reject, and session syslog messages. This feature simplifies integration with Security Information and Event Management (SIEM) systems that typically rely on the standard syslog format to aggregate event data across many different devices. The settings in this section enable and configure this feature.

For all of these settings, the argument is either **none** or a text string that includes references to event log variables. If the argument is **none**, then the corresponding event record is not written to the syslog file. This feature enables you to use the **syslog()** procedure in the policy without sending duplicate records to the syslog files. If the setting is not included in the **pb.settings** file, then Privilege Management for Unix and Linux performs syslog logging with hard-coded accept, reject, and session messages.

i The syslog logging feature must be enabled for customized syslog formatting to work. For more information, please see the following to enable syslog logging:

- "syslog" on page 119
- "facility" on page 120
- "syslogsessions" on page 138
- (-a and -r options) at "pbmasterd" on page 441
- (-a and -r options) at "pblogd" on page 439
- (-a option) at "pblockd" on page 430

To define a string to write to the syslog file, the entire text string must be enclosed in double quotation marks ("). An event log variable must be enclosed in percent character (%). A literal percent or double quotation mark character must be preceded by a back slash (\ and \%, for example). A particular item in a list variable can be referenced with the index number for that list (%argv[1]%, for example).



Note: This feature extends to one level of lists only; multi-level lists are not handled.

When an event that is recognized by one of these settings occurs, the text string is written to the syslog file, and the event log variable references are replaced with the values of those variables for that event. A variable reference that is not recognized is replaced with the string **<variable_name:undefined>** (**<variable_name [n]:undefined>** for unrecognized or nonexistent list items).



Note: Customized Syslog Formatting messages over 1,024 characters are truncated.



Note: When Privilege Management is installed, if a previous **pb.settings** file exists without the customized syslog formatting settings specified, then Privilege Management adds sample customized syslog formatting settings as comments. You can uncomment and remove the string **SAMPLE**, and then modify these sample settings.



For a list of event log variables, please see the [Privilege Management for Unix and Linux Language Guide](https://www.beyondtrust.com/docs/privilege-management/unix-linux/index.htm) at <https://www.beyondtrust.com/docs/privilege-management/unix-linux/index.htm>.

syslog_accept_format

- **Version 6.2 and earlier:** **syslog_accept_format** setting not available.
- **Version 7.0 and later:** **syslog_accept_format** setting available.

The **syslog_accept_format** setting defines the format of the record to be written to the syslog file for accept events.

**Example:**

```
syslog_accept_format "Privilege Management for Unix and Linux Master accepted
%command% on %date% at %hour%:%minute%. The command was submitted by
%user% on %submithost% and run by %runuser% on %runhost%"
```

Default

No default value

Used On

Policy server hosts

syslog_reject_format

- **Version 6.2 and earlier:** `syslog_reject_format` setting not available.
- **Version 7.0 and later:** `syslog_reject_format` setting available.

The `syslog_reject_format` setting defines the format of the record to be written to the syslog file for reject events.

**Example:**

```
syslog_reject_format "Privilege Management for Unix and Linux Master reject %command% on
%date% at %hour%:%minute%. The command was submitted by %user% on %submithost%"
```

Default

No default value

Used On

Policy server hosts

syslogsession_start_format

- **Version 6.2 and earlier:** `syslogsession_start_format` setting not available.
- **Version 7.0 and later:** `syslogsession_start_format` setting available.

The `syslogsession_start_format` setting defines the format of the record to be written to the syslog file for session start events.

**Example:**

```
syslog_accept_format "Privilege Management for Unix and Linux session started on
%date% at %hour%:%minute%. The session was started by %user%"
```

Default

No default value

Used On

Run hosts

syslogsession_start_fail_format

- **Version 6.2 and earlier:** `syslogsession_start_fail_format` setting not available.
- **Version 7.0 and later:** `syslogsession_start_fail_format` setting available.

The `syslogsession_start_fail_format` setting defines the format of the record to be written to the syslog file for session failed to start events.

**Example:**

```
syslogsession_start_fail_format "Privilege Management for Unix and Linux session failed
to start on %date% at %hour%:%minute%. User %user% attempted to start this session."
```

Default

No default value

Used On

Run hosts

syslogsession_finished_format

- **Version 6.2 and earlier:** `syslogsession_finished_format` setting not available.
- **Version 7.0 and later:** `syslogsession_finished_format` setting available.

The `syslogsession_finished_format` setting defines the format of the record to be written to the syslog file for session finished events.

**Example:**

```
syslogsession_finished_format "Privilege Management for Unix and Linux session finished on %date% at %hour%:%minute%. The session was started by %user%"
```

Default

No default value

Used On

Run hosts

syslogsession_finished_format_logserver

- **Version 10.0.1 and later:** `syslogsession_finished_format_logserver` setting available.

The `syslogsession_finished_format_logserver` setting defines the format of the record to be written to the syslog file for Finish events, from the log server.

**Example:**

```
syslogsession_finished_format "Task: '%runcommand%' finished at %exitdate% %exittime% as %runuser% on %runhost% with status %exitstatus%"
```

Default

No default value

Used On

Log servers

AD Bridge Event Logging

The BeyondTrust AD Bridge product enables you to attach Unix and Linux computers to a Microsoft Active Directory domain and manage them using Group Policies. Privilege Management for Unix and Linux can send information about certain events to AD Bridge for logging and reporting purposes. The settings in this section enable and configure the sending of event information to AD Bridge.

loadpbislibs

- **Version 6.2 and earlier:** `loadpbislibs` setting not available.
- **Version 7.0 and later:** `loadpbislibs` setting available.

The `loadpbislibs` setting loads the shared AD Bridge library at runtime, regardless of the value of the `pbis_event_logging` setting, provided that the `sharedlibpbisdependencies` setting is set with valid values and not set to **none**.



Example:

```
loadpbislibs yes
```

Default

```
loadpbislibs no
```

Used On

- Submit hosts
- Run hosts
- Policy server hosts
- Log hosts

pbis_event_logging

- **Version 6.2 and earlier:** `pbis_event_logging` setting not available.
- **Version 7.0 and later:** `pbis_event_logging` setting available.

The `pbis_event_logging` setting controls the writing of Privilege Management for Unix and Linux events (Accept, Reject, Finish, and Keystroke Action) to the AD Bridge event log database. A value of **yes** enables sending event information to AD Bridge and a value of **no** disables sending event information to AD Bridge. For this setting to have any effect, the `sharedlibpbisdependencies` setting must be set with valid values and not set to **none**.



Example:

```
pbis_event_logging yes
```

Default

```
pbis_event_logging no
```

Used On

- Policy server hosts
- Log hosts

pbis_log_connect_success

- **Version 6.2 and earlier:** `pbis_log_connect_success` setting not available.
- **Version 7.0 and later:** `pbis_log_connect_success` setting available.

The `pbis_log_connect_success` setting enables the sending of successful-connection events (to log hosts and policy server hosts) to the AD Bridge event log database. It also controls the posting of an event to AD Bridge if the policy server host is able to connect to the run host. For this setting to have any effect, the `sharedlibpbisdependencies` setting must be set with valid values and not set to `none`.



Example:

```
pbis_log_connect_success yes
```

Default

```
pbis_log_connect_success no
```

Used On

- Policy server hosts
- Submit hosts
- Run hosts

pbis_log_failover

- **Version 6.2 and earlier:** `pbis_log_failover` setting not available.
- **Version 7.0 and later:** `pbis_log_failover` setting available.

The `pbis_log_failover` setting enables the sending of policy server host and log host failover events to the AD Bridge event log database. It also controls the posting of an event to AD Bridge if the policy server host finds the run host unresponsive. For this setting to have any effect, the `sharedlibpbisdependencies` setting must be set with valid values and not set to `none`.

**Example:**

```
pbis_log_failover yes
```

Default

```
pbis_log_failover no
```

Used On

- Policy server hosts
- Submit hosts
- Run hosts

sharedlibpbisdependencies

- **Version 6.2 and earlier:** `sharedlibpbisdependencies` setting not available.
- **Version 7.0 and later:** `sharedlibpbisdependencies` setting available.

The `sharedlibpbisdependencies` setting specifies the shared libraries that are used to send event information to the AD Bridge event log. When set to `none` or when not set at all, no event information is sent to AD Bridge. If libraries are specified but they fail to load properly, Privilege Management for Unix and Linux records an error message and continues to operate without sending event information to AD Bridge.

**Example:**

```
sharedlibpbisdependencies  
/opt/pbis/lib/libeventlog_norpc.so.0.0.0 /opt/pbis/lib/liblwbbase_nothr.so.0.0.0
```

Default

The default value depends on the Privilege Management for Unix and Linux flavor and is determined at installation. You can change the value of this setting while running the Privilege Management for Unix and Linux installer or by modifying the `pb.settings` file after installation.

Used On

- Policy server hosts
- Log hosts
- Submit hosts
- Run hosts

BeyondInsight Event Logging

The BeyondTrustBeyondInsight product enables you to capture and report on privilege and vulnerability data across the entire IT stack (server, desktop, cloud, mobile, and virtualized environments).

Starting with version 7.5, Privilege Management for Unix and Linux can send information about certain events to BeyondInsight for logging and reporting purposes.

The settings in this section enable and configure the sending of event information to BeyondInsight.

 For more information, please see "[BeyondInsight Event and I/O Logging Common Settings](#)" on page 163.

rcshost

- **Version 7.1 and earlier:** `rcshost` setting not available.
- **Version 7.5 and later:** `rcshost` setting available.

The hostname of the Windows machine where BeyondInsight is installed. This keyword does not support the Privilege Management for Unix and Linux extended settings such as **interface**.

Example:

```
rcshost W7-RETINACS-01
```

Default

No default value

Used On

- Policy server hosts
- Log hosts

rcwebsvcport

- **Version 7.1 and earlier:** `rcwebsvcport` setting not available.
- **Version 7.5 and later:** `rcwebsvcport` setting available.

The port number used to communicate with BeyondInsight Web Services on **rcshost**.

Example:

```
rcwebsvcport 443
```

Default

```
rcswebsvcport 443
```

Used On

- Policy server hosts
- Log hosts

rctestworkgroup

- **Version 7.5 and earlier:** `rctestworkgroup` setting not available.
- **Version 8.0 and later:** `rctestworkgroup` setting available.

A label which helps BeyondInsight identify and sort data sent from Privilege Management for Unix and Linux.



Example:

```
rctestworkgroup PMULMasterBeyondTrustWorkgroup
```

Default

```
rctestworkgroup "BeyondTrust Workgroup"
```

Used On

Policy server hosts

sslrcscertfile

- **Version 7.1 and earlier:** `sslrcscertfile` setting not available.
- **Version 7.5 and later:** `sslrcscertfile` setting available.

BeyondInsight Client Certificate File in PEM format. Used to authenticate Privilege Management to BeyondInsight when sending event log records.



Example:

```
sslrcscertfile /etc/retinacs-01_eEyeEmsClient.pem
```

Default

No default value

Used On

- Policy server hosts
- Log hosts

sslrcscafile

- **Version 7.1 and earlier:** **sslrcscafile** setting not available.
- **Version 7.5 and later:** **sslrcscafile** setting available.

BeyondInsight server-bound Certificate Authority File in PEM format. Used to authenticate the BeyondInsight when sending event log records.



Example:

```
sslrcscafile /etc/retinacs-01_eEyeEmsCA.pem
```

Default

No default value

Used On

- Policy server hosts
- Log hosts

BeyondInsight I/O Log Indexing and Searching

Starting with version 7.5, Privilege Management can index I/O log files for an improved search capability using BeyondInsight Search GUI. Each log server and policy server host can communicate with a Solr server, submitting I/O log output data for indexing.

BeyondInsight provides a search GUI, allowing users to search indexed I/O logs.

The settings in this section enable and configure the indexing of I/O Log files with Solr.



For more information, please see "[BeyondInsight Event and I/O Logging Common Settings](#)" on page 163.

Solrhost

- **Version 7.1 and earlier:** Solrhost setting not available.
- **Version 7.5 and later:** Solrhost setting available.

The hostname where the Solr server is installed. This keyword does not support the Privilege Management for Unix and Linux extended settings such as **interface**.



Example:

```
Solrhost mySolrhost.mydomain
```

Default

No default value

Solrport

- **Version 7.1 and earlier:** Solrport setting not available.
- **Version 7.5 and later:** Solrport setting available.

The port number used to communicate with the Solr server.



Example:

```
Solrport 8443
```

Default

```
Solrport 8443
```

Used On

- Policy server hosts
- Log hosts

Solrvariables

- **Version 7.1 and earlier:** **Solrvariables** setting not available.
- **Version 7.5 and later:** **Solrvariables** setting available.

A list of Privilege Management for Unix and Linux policy variables, ending in **_pbul** that is used as stored data in Solr.



Example:

```
Solrvariables role_pbul list_pbul ticket_pbul
```

Default

No default value

Used On

- Policy server hosts
- Log hosts

Solrclientkeyfile

- **Version 7.1 and earlier:** **Solrclientkeyfile** setting not available.
- **Version 7.5 and later:** **Solrclientkeyfile** setting available.

Specifies a PEM format file containing the private key for a Solr client. The Solr server must be configured to have its Java keystore contain the Certificate Authority Certificate (CA cert) that signed the client's public certificate.



Example:

```
Solrclientkeyfile /etc/Solr.myhost.client.key.pem
```

Default

No default value

Used On

- Policy server hosts
- Log hosts

Solrclientcertfile

- **Version 7.1 and earlier:** Solrclientcertfile setting not available.
- **Version 7.5 and later:** Solrclientcertfile setting available.

Specifies a PEM format file containing the public certificate for the Solr client private key. The Solr server must be configured to have its Java keystore contain the Certificate Authority Certificate (CA cert) that signed the client's public certificate.



Example:

```
Solrclientcertfile /etc/Solr.myhost.client.cert.pem
```

Default

No default value

Used On

- Policy server hosts
- Log hosts

Solrcafile

- **Version 7.1 and earlier:** Solrcafile setting not available.
- **Version 7.5 and later:** Solrcafile setting available.

Specifies a PEM format file containing the Certificate Authority Certificate (CA cert) for the CA that signed the Solr server's SSL certificate. If this keyword is specified in **pb.settings**, **pbreplay** initiates an SSL connection to the Solr server. The **Solrport** keyword must be set to a port that Solr is using for HTTPS/SSL traffic.



Example:

```
Solrcafile /etc/Solr.myhost.ca.pem
```

Default

No default value

Used On

- Policy server hosts
- Log hosts

iologactiondb

- **Version 9.4.5 and earlier:** **iologactiondb** setting not available.
- **Version 10.0 and later:** **iologactiondb** setting available.

Optionally specifies the path and file name of a database used internally to schedule **iolog** indexing. This prevents too many **pbreplay** processes from overloading the system. If not specified, the default **pbiologaction.db** in the database directory is used.

Example:

```
iologactiondb /opt/pbul/dbs/action.db
```

Default

```
iologactiondb /opt/<prefix>pbul<suffix>/dbs/pbiologaction.db
```

Used On

- Policy server hosts
- Log hosts

iologactioninterval

- **Version 10.0.0 and earlier:** **iologactioninterval** not available.
- **Version 10.0.1 and later:** **iologactioninterval** available.

Optionally specifies the interval at which the scheduler checks to see if I/O logs need to be processed for Solr or **iologcloseactions**. The default is **60** seconds, and the minimum is **30** seconds.

Example:

```
iologactioninterval 120
```

Default

```
iologactioninterval 60
```

Used On

- Policy server hosts
- Log hosts

iologactionmaxprocs

- **Version 9.4.5 and earlier:** `iologactionmaxprocs` not available.
- **Version 10.0 and later:** `iologactionmaxprocs` available.

Optionally specifies a limit to the number of simultaneous **pbreplay** processes that can index I/O logs to Solr. This prevents too many **pbreplay** processes from overloading the system. If not specified, the default **pbiologaction.db** in the database directory is used.



Example:

```
iologactionmaxprocs 120
```

Default

```
iologactionmaxprocs 4
```

Used On

- Policy server hosts
- Log hosts

iologactionqueuetimelimit

- **Version 10.0.0 and earlier:** `iologactionqueuetimelimit` not available.
- **Version 10.0.1 and later:** `iologactionqueuetimelimit` available.

Optionally specifies the time limit, in minutes, that an **iolog** can be held in the processing queue without a heartbeat from **pblogd**, before that **iolog** is marked as ready for Solr or **iologcloseaction**. The default is **720** minutes (12 hours).


Example:

```
iologactionqueueetimit 300
```

Default

```
iologactionqueueetimit 720
```

Used On

- Policy server hosts
- Log hosts

iologactionqueue timeouts

- **Version 9.4.5 and earlier:** `iologactionqueue timeouts` available.
- **Version 10.0 and later:** `iologactionqueue timeouts` available.

The timeout values specified include:

- **[openread=timeout,delta,backoff]:** The overall timeout, the spin wait delta and the backoff modifier for the open for processing of `pblicense` write queues.
- **[openwrite=timeout,delta,backoff]:** The overall timeout, the spin wait delta and the backoff modifier for the open by clients to log transaction.
- **[write=timeout,delta,backoff]:** The overall timeout, the spin wait delta and the backoff modifier for waiting to write to the write queue.
- **[lock=timeout,delta,backoff]:** The overall timeout, the spin wait delta and the backoff modifier for waiting for exclusive lock when processing the `pblicense` write queues.


Example:

```
iologactionqueue timeouts openread=1000,10,2.0 openwrite=30000,5,1.2 write=30000,5,1.2 lock=30000,5,1.2
```

Default

No default value

Used On

- Policy server hosts
- Log hosts

iologactionretry

- **Version 10.0.0 and earlier:** `iologactionretry` not available.
- **Version 10.1.0 and later:** `iologactionretry` available.

Optionally specifies the interval, in minutes, that an **iolog** must wait for a Solr or **iologcloseaction** retry. A Solr attempt is requeued in certain recoverable cases, such as when unable to reach the host. An **iologcloseaction** attempt is requeued if the **iologcloseaction** script returns **-1**. This delay allows time for the issue to be potentially resolved before the next attempt. The minimum is **5** minutes and the maximum is **2880** (48 hours).



Example:

```
iologactionretry 300
```

Default

```
iologactionretry 20
```

Used On

- Policy server hosts
- Log hosts

iologindexstorefile

- **Version 7.1 and earlier:** `iologindexstorefile` setting not available.
- **Version 7.5 through 9.4.5 all OS:** `iologindexstorefile` setting available.
- **Version 10.1.0 and later:** `iologindexstorefile` setting not available.

The path and file name of the file used to store I/O log file names that failed to be forwarded to Solr due to an error. This file is periodically scanned by **pblogd** and the content forwarded to Solr, when the communication with Solr is reestablished.



Example:

```
iologindexstorefile /var/log/pb.iolog.store
```

Default

```
iologindexstorefile <default_log_directory>/pb.iolog.store
```

Used On

- Policy server hosts
- Log hosts

indexcommandtimestamps

- **Version 7.1 and earlier:** `indexcommandtimestamps` setting not available.
- **Version 7.5 and later:** `indexcommandtimestamps` setting available.

Used to disable command timestamps in the Solr index. Command timestamps in the Solr index can be used to search for commands that happened near a time. These timestamps are enabled by default.

Example:

```
indexcommandtimestamps no
```

Default

```
indexcommandtimestamps yes
```

Used On

- Policy server hosts
- Log hosts

indexlogsize-limit

- **Version 9.4.5 and earlier:** `indexlogsize-limit` setting not available.
- **Version 10.0 and later:** `indexlogsize-limit` setting available.

Used to set a size limit for I/O logs that can be indexed. The `indexlogsize-limit` keyword is an integer optionally followed by **k|K|m|M|g|G**. Any additional characters are ignored.

Example:

```
indexlogsize-limit 60M
```

Default

No default value

Used On

- Policy server hosts
- Log hosts

pbreplaylog

- **Version 7.1 and earlier:** **pbreplaylog** setting not available.
- **Version 7.5 and later:** **pbreplaylog** setting available .

pbreplaylog contains the name for **pbreplay**'s diagnostic log file.



Example:

```
pbreplaylog /var/log/pbreplay.log
```

Default

During the install, depending on the operating system standards, this can be any of the following:

```
pbreplaylog /var/log/pbreplay.log
```

```
pbreplaylog /usr/log/pbreplay.log
```

```
pbreplaylog /var/adm/pbreplay.log
```

```
pbreplaylog /usr/adm/pbreplay.log
```

Used On

- Policy server hosts
- Log hosts
- GUI hosts

Solrindextimeout

- **Version 9.4.5 and earlier:** **Solrindextimeout** setting not available.
- **Version 10.0 and later:** **Solrindextimeout** setting available.

Used to set a time limit for I/O logs being indexed. If Solr indexing exceeds the specified time limit, indexing the current **iolog** is terminated. The time limit, specified in seconds, takes place for both the connection phase and the sending of each 5MB chunk to Solr. For example, if **Solrindextimeout** is set to 15, an **iolog** with 10MB **stdout** data might take up to 60 seconds connecting and talking to Solr before timing out. If **Solrindextimeout** is not set, or is set to **-1**, there is no timeout.

**Example:**

```
Solrindextimeout 120
```

Default

```
Solrindextimeout -1
```

Used On

- Policy server hosts
- Log hosts

BeyondInsight Event and I/O Logging Common Settings



Note: The settings in this section must be set, in addition to the settings described in "BeyondInsight Event Logging" on page 150 and "BeyondInsight I/O Log Indexing and Searching" on page 153.

pbadminpath

- **Version 7.1 and earlier:** pbadminpath setting not available.
- **Version 7.5 and later:** pbadminpath setting available.

The path where admin binaries reside. This is used by **pbmasterd** and **pblogd** to forward events and iologs via **pbreplay**.



Example:

```
pbadminpath /usr/sbin/
```

Default

No default value

Used On

- Policy server hosts
- Log hosts
- GUI hosts

guiport

- **Version 7.1 and earlier:** guiport setting not available.
- **Version 7.5 and later:** guiport setting available.

Defines the TCP port number that is used for PBGUID. The port numbers for Privilege Management daemons must use the non-reserved system ports. The allowed port numbers are **1024** to **65535** (inclusive).

In a BeyondInsight integrated environment, this port number is used by the BeyondInsight Console to replay I/O logs.



Example:

```
guiport 24348
```

Default

```
guiport 24348
```

Used On

- Policy server hosts
- Log hosts
- GUI hosts

sguiport

- **Version 7.1 and earlier:** `sguiport` setting not available.
- **Version 7.5 and later:** `sguiport` setting available.

Defines the TCP port number that is used for the secure PBSGUID. The port numbers for Privilege Management daemons must use the non-reserved system ports. The allowed port numbers are **1024 to 65535** (inclusive).

In a BeyondInsight integrated and SSL-enabled environment, this secure port number is used by the BeyondInsight Console to replay I/O logs.



Example:

```
sguiport 24349
```

Default

```
sguiport 24349
```

Used On

- Policy server hosts
- Log hosts
- GUI hosts

sharedlibcurldependencies

- **Version 7.1 and earlier:** `sharedlibcurldependencies` setting not available.
- **Version 7.5 and later:** `sharedlibcurldependencies` setting available.

Defines the path and file name for `libcurl` used by the binaries to communicate with Solr server and/or BeyondInsight server. Setting `sharedlibcurldependencies` requires `sharedlibkrb5dependencies` and `sharedlibssldependencies` to be set.



Note: The certificates used for BeyondInsight by Privilege Management for Unix & Linux need support for SHA-256 algorithm which was introduced in SSL v0.9.8. Therefore starting with v7.5, the SSL libraries shipped with Privilege Management for Unix and Linux Shared Libraries are SSL libraries v0.9.8. When installing or upgrading, **sharedlibssldependencies** need to be set to:

```
/usr/lib/beyondtrust/pb/libcrypto.so.1.1 /usr/lib/beyondtrust/pb/libssl.so.1.1
```



Example:

```
sharedlibcurldependencies /usr/lib/beyondtrust/pb/libcurl.so.4.7.0"
```

Default

- AIX: /usr/lib/symark/pb/libcurl.a(libcurl.so.4)
- HPUX Itanium: /usr/lib/symark/pb/libcurl.so.7.0
- Linux, Solaris: /usr/lib/symark/pb/libcurl.so.4.3.0

Used On

- Policy server hosts
- Log hosts

loadcurllibs

- **Version 7.5 and earlier:** loadcurllibs setting not available.
- **Version 7.5 and later:** loadcurllibs setting available.

Forces the loading of **libcurl** libraries defined in **sharelibcurldependencies**.



Example:

```
loadcurllibs yes
```

Default

```
loadcurllibs no
```

Used On

- Policy server hosts
- Log hosts

Event Queueing of Integrated Products

Privilege Management for Unix and Linux sends its events to other products, such as BeyondTrustBeyondInsight, for additional processing to provide added value to users. Starting with v10.1.0, a message router and event scheduler is used to manage the transmission of event data to the integrated products. This architecture comes with additional configuration settings.

integratedproductsqueuedb

- **Version 10.0.0 and earlier:** `integratedproductsqueuedb` setting not available.
- **Version 10.1.0 and later:** `integratedproductsqueuedb` setting available.

The `integratedproductsqueuedb` setting specifies the path to the event queue database that holds the event data to be forwarded by the scheduler to the integrated product(s). If a relative path is specified, the `basedir` setting is used to derive the full path.



Example:

```
integratedproductsqueuedb /mypath/pbipevent.db
```

Default

```
integratedproductsqueuedb /opt/<prefix>pbul<suffix>/dbs/pbintprodq.db
```

Used On

- Log hosts
- Policy server hosts

autofwdtime

- **Version 7.1 and earlier:** `autofwdtime` setting not available.
- **Version 7.5 and later:** `autofwdtime` setting available.
- **Version 10.1.0 and later:** `autofwdtime` setting available, but usage modified.

The interval, in minutes, that defines how often the scheduler processes the queued events to be forwarded to integrated products (BeyondInsight, for example). If this keyword does not exist in the settings file, it defaults to **20** minutes.



Example:

```
autofwdtime 4
```

Default

```
autofwdtime 20
```

Used On

- Policy server hosts
- Log hosts

Log Synchronization

Beginning with v5.0, Privilege Management for Unix and Linux can consolidate and merge the logs from the log server and the secondary server following a log server failover and log server recovery. The log synchronization feature is controlled by the **logresynctimermin**, **pbsyncdlog**, **pbsynclog**, and **syncport** settings. The client also uses log servers and event logs.

logresynctimermin

- **Version 4.0 and earlier:** **logresynctimermin** setting not available.
- **Version 5.0 and later:** **logresynctimermin** setting available.

When **pbsync** is started in daemon mode, this variable defines how often the client attempts to resynchronize the files. The time is defined in minutes and can be 5 minutes or greater.



Example:

```
logresynctimermin 15
```

Default

```
logresynctimermin 16
```

Used on

- Log hosts
- Synchronization clients

pbsyncdlog

- **Version 4.0 and earlier:** **pbsyncdlog** setting not available.
- **Version 5.0 and later:** **pbsyncdlog** setting available.

There will be a unique file to keep track of server transactions. This variable refers to the path and file name for this feature's log, typically:

```
/path_to_logs/pbsyncd.log
```



Example:

```
pbsyncdlog /var/log/pbsyncd.log
```

Default

During the install, depending on the operating system standards, this can be any of the following:

```
pbsyncdlog /var/log/pbsyncd.log
```

```
pbsyncdlog /usr/log/pbsyncd.log
```

```
pbsyncdlog /var/adm/pbsyncd.log
```

```
pbsyncdlog /usr/adm/pbsyncd.log
```

Used on

- Log hosts
- Synchronization clients

pbsynclog

- **Version 4.0 and earlier:** **pbsynclog** setting not available.
- **Version 5.0 and later:** **pbsynclog** setting available.

There is a unique file to keep track of client transactions. **pbsynclog** refers to the path and file name for this feature's log, typically:

```
/path_to_logs/pbsync.log
```



Example:

```
pbsynclog /var/log/pbsync.log
```

Default

During the install, depending on the operating system standards, this can be any of the following:

```
pbsynclog /var/log/pbsync.log
```

```
pbsynclog /usr/log/pbsync.log
```

```
pbsynclog /var/adm/pbsync.log
```

```
pbsynclog /usr/adm/pbsync.log
```

Used on

- Log hosts
- Synchronization clients

syncport

- **Version 4.0 and earlier:** `syncport` setting not available.
- **Version 5.0 and later:** `syncport` setting available.

`syncport` defines the TCP port number that is used for log synchronization. The port numbers for Privilege Management daemons must use the non-reserved system ports. The allowed port numbers are **1024 to 65535** (inclusive).



Example:

```
syncport 24350
```

Default

```
syncport 24350
```

Used on

- Log hosts
- Synchronization clients

syncprotocoltimeout

- **Version 5.2 and earlier:** `syncprotocoltimeout` setting not available.
- **Version 6.0 and later:** `syncprotocoltimeout` setting available.

After a connection is established between a log synchronization client (**pbsync**) and server (**pbsyncd**), the programs perform protocol checks to verify a proper connection. Some types of protocol failures could take a long time to determine (for example., wrong service running on the policy server port, or mismatched encryption types/keys).

The `syncprotocoltimeout` setting determines the maximum time to wait for protocol completion. If a protocol step does not complete within the specified number of milliseconds, then **pbsync** stops with an error. A value of **-1** indicates no protocol timeout.

**Example:**

```
syncprotocoltimeout 2000
```

Default

```
syncprotocoltimeout -1
```

Used on

- Log hosts
- Log synchronization hosts

Diagnostic Logging

pblogd, **pbrun**, **pbmaterd**, **pblocald**, **pblogd**, **pbksh**, **pbsh**, and **pbguid** can produce diagnostic messages that can be stored in individual files.

kshlog

- **Version 3.5 and earlier:** **kshlog** setting not available.
- **Version 4.0 and later:** **kshlog** setting available.

kshlog contains the name of the **pbksh** diagnostic log file.



Example:

```
kshlog /var/log/pbksh.log
```

Default

During the install, depending on the operating system standards, this can be any of the following:

```
kshlog /var/log/pbksh.log
```

```
kshlog /usr/log/pbksh.log
```

```
kshlog /var/adm/pbksh.log
```

```
kshlog /usr/adm/pbksh.log
```

Used on

Submit hosts

pbguidlog

- **Version 4.0.0 and later:** **pbguidlog** setting available.

pbguidlog contains the name of the **pbguid** diagnostic log file.



Example:

```
pbguidlog /var/log/pbguid.log
```

Default

During the install, depending on the operating system standards, this can be any of the following:

```
pbguidlog /var/log/pbguid.log
```

```
pbguidlog /usr/log/pbguid.log
```

```
pbguidlog /var/adm/pbguid.log
```

```
pbguidlog /usr/adm/pbguid.log
```

Used on

GUI hosts

pblocaldlog

- **Version 4.0.0 and later:** **pblocaldlog** setting available.

pblocaldlog contains the name of the **pblocald** diagnostic log file.



Note: This keyword does not apply to **pbssh**. If it is present in the settings file, it does not have any effect on **pbssh** and is ignored.



Example:

```
pblocaldlog /var/log/pblocald.log
```

Default

During the install, depending on the operating system standards, this can be any of the following:

```
pblocaldlog /var/log/pblocald.log
```

```
pblocaldlog /usr/log/pblocald.log
```

```
pblocaldlog /var/adm/pblocald.log
```

```
pblocaldlog /usr/adm/pblocald.log
```

Used on

Run hosts

pblogdlog

- **Version 4.0.0 and later:** **pblogdlog** setting available.

pblogdlog contains the name of the **pblogd** diagnostic log file.



Example:

```
pblogdlog /var/log/pblogd.log
```

Default

During the install, depending on the operating system standards, this can be any of the following:

```
pblogdlog /var/log/pblogd.log
```

```
pblogdlog /usr/log/pblogd.log
```

```
pblogdlog /var/adm/pblogd.log
```

```
pblogdlog /usr/adm/pblogd.log
```

Used on

Log hosts

pbmasterdlog

- **Version 4.0.0 and later:** **pbmasterdlog** setting available.

pbmasterdlog contains the name of the **pbmasterd** diagnostic log file.



Example:

```
pbmasterdlog /var/log/pbmasterd.log
```

Default

During the install, depending on the operating system standards, this can be any of the following:

```
pbmasterdlog /var/log/pbmasterd.log
```

```
pbmasterdlog /usr/log/pbmasterd.log
```

```
pbmasterdlog /var/adm/pbmasterd.log
```

```
pbmasterdlog /usr/adm/pbmasterd.log
```

Used on

Policy server hosts

pbrunlog

- **Version 4.0.0 and later:** **pbrunlog** setting available.

pbrunlog contains the name of the **pbrun** diagnostic log file.



Example:

```
pbrunlog /var/log/pbrun.log
```

Default

No default value

Used on

Submit hosts

pbpinglog

- **Version 6.2 and earlier:** **pbpinglog** setting not available.
- **Version 7.0 and later:** **pbpinglog** setting available.

pbpinglog contains the name of the **pbping** diagnostic log file.

**Example:**

```
pbpinglog /var/log/pbping.log
```

Default

No default value

Used on

Policy server hosts

shlog

- **Version 3.5 and earlier:** shlog setting not available.
- **Version 4.0 and later:** shlog setting available.

shlog contains the name of the **pbsh** diagnostic log file.

**Example:**

```
shlog /var/log/pbsh.log
```

Default

During the install, depending on the operating system standards, this can be any of the following:

```
shlog /var/log/pbsh.log
```

```
shlog /usr/log/pbsh.log
```

```
shlog /var/adm/pbsh.log
```

```
shlog /usr/adm/pbsh.log
```

Used on

Submit hosts

enabletraceonexit

- **Version 22.2.0 and later:** `enabletraceonexit` setting available.

Controls extended diagnostic logging when certain error conditions are encountered.

This should only be used in coordination with BeyondTrust Technical Support.

This is disabled (set to no) by default.

Example

```
enabletraceonexit yes
```

Default

no

Used on

All PMUL hosts

pbadminlog

- **Version 22.1 and earlier:** `pbadminlog` setting not available.
- **Version 22.2 and later:** `pbadminlog` available.

The `pbadminlog` setting defines the pathname of the log file containing any diagnostics messages generated when running the `pbadmin` program.



Note: *pbadmin* is the same program as *pbdbutil*, which may still be referenced in this manual.

The `pbadmin` program is used during the installation of PMUL. Immediately after running `pbinstall` for the first time on a host, you may find this log entry which can be ignored:

```
3887.22 Error retrieving license - Failed to retrieve your license
```

Example

Depending on the operating system standards, this can be any of the following:

```
/var/log/<prefix>pbadmin.log<suffix>
```

```
/var/adm/<prefix>pbadmin.log<suffix>
```

```
/usr/adm/<prefix>pbadm.in.log<suffix>
```

Default

```
pbadm.inlog /var/log/pbadm.in.log
```

Used On

All hosts

I/O Log Action

iologactionwq

- The **iologactionwq** setting is available only in versions **10.0.0** and **10.0.1**.

The **iologactionwq** keyword defines the initial name of the write queue files for **iologcloseaction** and Solr. This file is created in **basedir** by default, unless the file name starts with 'l'.



Example:

```
iologactionwq "pbilogaction.wq"
```

Default

No default value

Used On

- Log servers
- Policy servers

iologactionwqnum

- The **iologactionwqnum** setting is available only in versions **10.0.0** and **10.0.1**.

The **iologactionwqnum** keyword defines the number of write queue files for **iologcloseaction** and Solr. Generally, the more files allowed, the fewer file locks will be encountered and thus increase speed. The maximum value is **999**.



Example:

```
iologactionwqnum 99
```

Default

```
iologactionwqnum 10
```

Used On

- Log servers
- Policy servers

iologactiondb

- **Version 10.0.1 and later:** `iologactiondb` setting available.

The `iologactiondb` keyword specifies the database filename for the `iologcloseaction` and Solr queue. This file is created in `basedir` by default, unless the file name starts with a slash (`/`).



Example:

```
iologactiondb pbiologaction.db
```

Default

```
iologactiondb /opt/<prefix>pbul<suffix>/dbs/pbiologaction.db
```

Used On

- Log servers
- Policy servers

iologactiondbdelay

- **Version 10.0.1 and later:** `iologactiondbdelay` available.

When opening the `iologaction` database, the `iologactiondbdelay` keyword specifies the number of milliseconds to wait when the `iologaction` database is locked.



Example:

```
iologactiondbdelay 10000
```

Default

```
iologactiondbdelay 10000
```

Used On

- Log servers
- Policy servers

iologactionmaxprocs

- **Version 10.0.1 and later:** `iologactionmaxprocs` setting available.

The `iologactionmaxprocs` keyword specifies the maximum number of `pbreplay` processes used to process Solr indexing and `iologcloseaction` actions. Limiting the maximum number of processes reduces CPU load used for Solr indexing and `iologcloseaction` at any given time.



Example:

```
iologactionmaxprocs 20
```

Default

```
iologactionmaxprocs 4
```

Used On

- Log servers
- Policy servers

iologactionretry

- **Version 10.0.1 and later:** `iologactionretry` setting available.

The `iologactionretry` keyword specifies the time, in minutes, before it a requeued task is processed. The minimum value allowed is **5**; the maximum is **2880** (48 hours).



Example:

```
iologactionretry 60
```

Default

```
iologactionretry 20
```

Used On

- Log servers
- Policy servers

Log Archiving

Beginning with v9.0, Privilege Management for Unix and Linux provides a logfile tracking and archiving mechanism for I/O logs and eventlogs.

enablelogtrackingdb

- **Version 8.5 and earlier:** `enablelogtrackingdb` setting not available.
- **Version 9.0 and later:** `enablelogtrackingdb` setting available.

For use on log hosts and policy server hosts. If set, the Privilege Management for Unix and Linux component creating the event log or I/O log sends the location information to the centralized tracking database to be recorded. This setting requires a configured REST service on the designated Log Archiver Database Server, and needs `logarchivedbhost` and `pbrestport` settings in order to update the database. To disable the feature, set this to `no`, and the log writer will not send the logfile location to the log tracking database. It is enabled by default.



Example: Enable Tracking of Logfile Location:

```
enablelogtrackingdb yes
```



Example: Disable Tracking of Logfile Location:

```
enablelogtrackingdb no
```

Default

```
enablelogtrackingdb no
```

Used on

- Log hosts
- Policy server hosts if a log host is not used

logarchivehost

- **Version 8.5 and earlier:** `logarchivehost` setting not available.
- **Version 9.0 and later:** `logarchivehost` setting available.

For use on log servers where the logfile originates. It is the name of the default destination host that receives the archived log files. It must have a valid Privilege Management installation with the REST service configured.

**Example:**

```
logarchivehost host
```

host is the hostname or IP address of the archive host.

Default

No default value

Used on

- Log hosts
- Policy server hosts if a log host is not used

logarchivedbhost

- **Version 8.5 and earlier:** logarchivedbhost setting not available.
- **Version 9.0 and later:** logarchivedbhost setting available.

For use on log servers where the logfile originates. It is the name or the IP address of the host where the log tracking database is created and maintained.



Note: The host specified must have a valid Privilege Management installation with the REST service configured.

**Example:**

```
logarchivedbhost logarchdbhost1  
logarchivedbhost 192.10.42.235
```

Default

No default value

Used on

- Log hosts
- Policy server hosts if a log host is not used

logarchivedir

- **Version 8.5 and earlier:** `logarchivedir` setting not available.
- **Version 9.0 and later:** `logarchivedir` setting available.

It defines the main destination path for the log files on the Log Archive Storage Server host. Under this main directory, the logfiles are organized appropriately in their subdirectories:

- event logs: `<logarchivedir>/eventlog/<origlogservername>`
- I/O logs: `<logarchivedir>/iolog/<submithost>/submituser/<date>`



Note: If the directory does not yet exist, it is created and made secure (readable and writable by root only).



Example:

```
logarchivedir /pbul/pbarchive
```

Default

During the install, depending on the operating system standards, this can be any of the following:

```
logarchivedir /var/log/pblogarchive
```

```
logarchivedir /usr/log/pblogarchive
```

```
logarchivedir /var/adm/pblogarchive
```

```
logarchivedir /usr/adm/pblogarchive
```

Used on

Log hosts designated as Log Archive Storage Server

logarchivedb

- **Version 8.5 and earlier:** `logarchivedb` setting not available.
- **Version 9.0 and later:** `logarchivedb` setting available.

The absolute path of the SQLite log tracking database file on the Log Archiver Database Server. If the file does not yet exist, it is created when the first row is inserted.


Example:

```
logarchivedb /var/log/pblogtrack.db
```

Default

```
logarchivedb /opt/<prefix>dbs<suffix>/dbs/pblogarchive.db
```

Used on

Log hosts designated as Log Archiver Database Server

logarchivedb_delay

- **Version 9.4.0 and earlier:** `logarchivedb_delay` setting not available.
- **Version 9.4.1 and later:** `logarchivedb_delay` setting available.

Maximum accumulated time in milliseconds that the log host busy handler sleeps during the retry cycle when it encounters a locked log tracking database. If not specified, the default value is **100,000** milliseconds. The valid range is **0 - 1,200,000** milliseconds. A **0** value means no retries are attempted and a *database locked* error is logged immediately. Increase the value if there is a high demand on updating the log tracking database and there are too many database locked errors reported. A higher value, however, may affect the performance of the log host.



Note: SQLite may not invoke the busy handler if it determines the possibility of a deadlock.


Example:

```
logarchivedb_delay 200000
```

Default

```
logarchivedb_delay 10000
```

Used on

Log hosts designated as Log Archiver Database Server

Logfile Name Caching

Beginning with v9.4.0, if you are integrating with the BeyondInsight for Unix & Linux, you can enable the caching of logfile names. The logfile names, creation/access time, and run host association are saved in SQLite databases for better console experience and functionality.

enablelogcaching

- **Version 10.3.0 and earlier:** `enablelogcaching` setting not available.
- **Version 10.3.1 and later:** `enablelogcaching` setting available.

When set to **yes**, log file name caching is enabled and the Privilege Management for Unix and Linux services refer to `logcachedb` and `iologcachedb` settings keywords for the appropriate database paths. Set to **no** to disable log file name caching feature. Default value is **yes** (if commented out or omitted from the settings file).

logcachedb

- **Version 9.3 and earlier:** `logcachedb` setting not available.
- **Version 9.4 and later:** `logcachedb` setting available, with the following behavior:

This setting holds the absolute path of the SQLite database file which stores information on the event log and I/O log file pathnames on the log host. If the database file does not yet exist, it is created when the first row is inserted.



Note: If this setting does not exist in the `pb.settings` file, it disables the log file name caching feature.

- **Version 10.3.1 and later:** `logcachedb` setting available, with the following behavior:

Starting in v10.3.1, this setting holds the absolute path of the SQLite database file which stores information only on Privilege Management for Unix and Linux event log file pathnames on the log host. If the database file does not yet exist, it is created when the first row is inserted.



Note: Commenting out this setting does not disable log file name caching feature. To enable/disable log file name caching feature, use the `enablelogcaching` setting.



Example:

```
logcachedb /var/log/pblogcache.db
```

Default

```
logcachedb /opt/<prefix>pbul<suffix>/dbs/pblogcache.db
```

Used on

Log hosts

logcachedb_delay

- **Version 9.4.0 and earlier:** `logcachedb_delay` setting not available.
- **Version 9.4.1 and later:** `logcachedb_delay` setting available.

Maximum accumulated time in milliseconds that the log host busy handler sleeps during the retry cycles when it encounters a locked `logcachedb` database. If not specified, the default value is **100,000** milliseconds. The valid range is **0 - 1,200,000** milliseconds. A **0** value means no retries are attempted and a *database locked* error is logged immediately. Increase the value if there is a high demand on updating the log file name caching database and there are too many database locked errors reported. A higher value, however, may affect the performance of the log host.



Note: SQLite may decide not to invoke the busy handler if it determines the possibility of a deadlock.



Example:

```
logcachedb_delay 200000
```

Default

```
logcachedb_delay 10000
```

Used on

Log hosts

iologcachedb

- **Version 10.3.0 and earlier:** `iologcachedb` setting not available.
- **Version 10.3.1 and later:** `iologcachedb` setting available.

This setting holds the absolute path of the SQLite database file which stores information only on Privilege Management for Unix and Linux I/O log file pathnames on the log host. If the database file does not yet exist, it is created when the first row is inserted.



Note: Commenting out this setting does not disable the log file name caching feature. To enable/disable the log file name caching feature, use the `enablelogcaching` setting.

**Example:**

```
iologcachedb /var/log/pbiologcache.db
```

Default

```
iologcachedb /opt/<prefix>pbul<suffix>/dbs/pbiologcache.db
```

Used on

Log hosts

iologcachedb_delay

- **Version 10.3.0 and earlier:** `iologcachedb_delay` setting not available.
- **Version 10.3.1 and later:** `iologcachedb_delay` setting available.

Maximum accumulated time in milliseconds that the log host busy handler sleeps during the retry cycles when it encounters a locked `iologcachedb` database. If not specified, the default value is **100,000** milliseconds. The valid range is **0 - 1,200,000** milliseconds. A **0** value means no retries are attempted and a database locked error is logged immediately. Increase the value if there is a high demand on updating the log file name caching database and there are too many database locked errors reported. A higher value, however, may affect the performance of the log host.



Note: *SQLite may decide not to invoke the busy handler if it determines the possibility of a deadlock.*

**Example:**

```
iologcachedb_delay 20000
```

Default

```
iologcachedb_delay 10000
```


Used on

Log hosts

Network Traffic and File Encryption

Privilege Management for Unix and Linux can encrypt network traffic, event logs, I/O logs, policy files, and its own settings file. The following table lists the available encryption algorithms. If you are using SSL, then it supersedes the network traffic encryption algorithms after the start-up protocol is complete.

Encryption Algorithms

Settings String	Block Size (bytes)	Key Size (bytes)	Comments
3des	8	24	Old style Triple DES. This algorithm is deprecated in favor of the new style Triple DES and will be removed in a future release.
aes-16-16 (or aes-128)	16	16	AES
aes-16-24 (or aes-192)	16	24	" "
aes-16-32 (or aes-256)	16	32	" "
aes-24-16	24	16	" "
aes-24-24	24	24	" "
aes-24-32	24	32	" "
aes-32-16	32	16	" "
aes-32-24	32	24	" "
aes-32-32	32	32	" "
blowfish	8	56	Blowfish
cast128	8	16	Cast-128
des	8	8	DES
gost	8	32	Gost
loki97	16	32	Loki97
none	0	0	No encryption.
old	stream	1024	<p>A proprietary algorithm, maintained for backward compatibility only.</p> <div style="border: 1px solid black; padding: 5px; background-color: #e0f0ff;">  Note: This algorithm is deprecated and will be removed in a future release. </div>

Settings String	Block Size (bytes)	Key Size (bytes)	Comments
saferplus-16	16	16	SaferPlus
saferplus-24	16	24	" "
saferplus-32	16	32	" "
serpent-16	16	16	Serpent
serpent-24	16	24	" "
serpent-32	16	32	" "
threeway	12	12	Threeway
tiny	8	16	Tiny
tripleDES	8	16	New style Triple DES
twofish-16	16	16	Twofish
twofish-24	16	24	" "
twofish-32	16	32	" "

Enhanced Encryption

To enable compliance with US government regulations, and specifically FIPS 140-2, encryption has been updated. Many of the older less secure encryption algorithms have been deprecated, and when high security is enforced, they are disabled completely.

When new clients are installed, **enforcehighsecurity** and **ssl** are both enabled in **pb.settings**. This switches Privilege Management for Unix and Linux into FIPS 140-2 mode. All encryption algorithms are FIPS 140-2 compliant, and it does not communicate, encrypt, or decrypt any data that isn't encrypted in AES-128, AES-192, AES-256 or TripleDes (3DES).



Note: If a customer is installing version 9 of Privilege Management for Unix and Linux from scratch, high security mode is recommended.

For existing customers who are upgrading their enterprise to version 9, the upgrade script automatically adds the AES-256 encryption algorithm onto the I/O log and event log encryption configuration, leaving the existing encryption algorithms at the end of the configuration. This ensures that new I/O logs and event logs are encrypted using modern secure algorithms, but allows existing I/O logs and event logs that are encrypted in less secure algorithms to be decrypted and retrieved. Although existing network encryption can continue to use deprecated encryption algorithms, because the data is transient, more permanent data such as I/O logs and event logs can only be encrypted in FIPS 140-2 compatible algorithms.



Note: Customers who have an existing infrastructure, and would like to be FIPS 140-2 compliant must upgrade all Privilege Management for Unix and Linux servers and clients to the latest version. If there are existing I/O logs and event logs that are encrypted using less secure algorithms, a specially configured host is required that is dedicated to reading these older logs.

To accomplish this task, you can use the new Client Registration feature to copy new **pb.settings** configuration, keys and certificates, or you can configure each installation by hand and copy the files manually.

Use Client Registration

1. Follow the upgrade guide to update the primary policy server to the latest version.
2. Create a new-style encryption key to be used across the enterprise:

```
pbkey -F /etc/pbfips.key
```

3. Create a suitable client **pb.settings** file, for example **/etc/pb-client.settings**, and configure the new encryption settings.



Example:

```
enforcehighsecurity yes
ssl yes
ssloptions requiressl    sslfirst  sslverbose
sslservercertfile /etc/pbmasterhost.crt
sslserverkeyfile /etc/pbmasterhost.pem
networkencryption aes-256:keyfile=/etc/pbfips.key
iologencryption aes-256:keyfile=/etc/pbfips.key
eventlogencryption aes-256:keyfile=/etc/pbfips.key
submitmasters pbmasterhost.org.com
logservers pbloghost.org.com
```

4. Follow the Client Registration guide to enable the service and configure an appropriate client profile.



Example: For example, on the primary policy server run:

```
pbdbutil --reg -n
pbdbutil --reg -u '{"name":"client-prof","data":
[{"type":"settings","fname":"/etc/pb-client.settings"},
{"type":"certificate","to":"/etc/${prefix}pbrest.pem${suffix}"},
{"type":"save","sname":"networkencryption"},
{"type":"save","sname":"iologencryption"},
{"type":"save","sname":"eventlogencryption"},
{"type":"save","sname":"restkeyencryption"},
{"type":"save","sname":"sslservercertfile"},
{"type":"save","sname":"sslserverkeyfile"}]}'
```

5. Create similar profiles for your secondary policy servers, log servers, etc.
6. Create a REST application ID and Key to authenticate your Client Registration requests.



Example: For example, on the primary policy server run:

```
pbdbutil --rest -g clientreg
{"appkey":"cbbc1aab-6f2b-40d0-b611-060bff0aaafa"}
```


- Now follow the upgrade guide to upgrade each client and server, using Client Registration when prompted. Run the normal **pbinstall** on the client and when asked whether to use Client Registration, answer **yes**, and provide responses to the Client Registration configuration questions.


Example:

```
Do you wish to utilize Client Registration? [yes]?
Enter the Application ID generated on the Primary License Server: clientreg
Enter the Application Key generated on the Primary License Server: cbbclaab-
6f2b-40d0-b611-060bff0aaafa
Enter the Primary License Server address/domain name for registering
clients: pbmasterhost.org.com
Enter the Primary License Server REST TCP/IP port [24351]: 24351
Enter the Registration Client Profile name [default]: client-prof
```

Using the profile appropriate to the installation type. All the necessary **pb.settings**, keys, and certificates are automatically copied to the upgrade installation, making upgrade simple.

Alternatively, these Client Registrations options can be specified on the **pbinstall** command line for automation.


Example:

```
pbinstall -A clientreg -K cbbclaab-6f2b-40d0-b611-060bff0aaafa -D
pbmasterhost.org.com -N client-prof
```

Without Using Client Registration

- Follow the upgrade guide to update the primary policy server to the latest version.
- Create a new-style encryption key to be used across the enterprise:

```
pbkey -F /etc/pbfips.key
```

- For each upgrade you need to copy the new key, the primary policy manager certificate and the primary policy server key to each host.
- During upgrade you need to change settings to enable high security mode:

```
Enforce High Security Encryption? yes Use SSL? yes
SSL Configuration? requiresslsslfirst sslverbose
SSLServer Certificate File? <path to Primary Policy Server certificate file>
SSL Server Private Key File? <path to Primary Policy Server key file>
PowerBroker network encryption options aes-256:keyfile=/etc/pbfips.key
PowerBroker event log encryption options aes-256:keyfile=/etc/pbfips.key
PowerBroker I/O log encryption options aes-256:keyfile=/etc/pbfips.key
```

To configure a dedicated host to read older I/O logs and event logs encrypted with deprecated encryption algorithms, the following configuration is required to ensure that it can communicate with the new FIPS 140-2 compliant installations, but allowing it to read the older logs. Follow the above installation procedures, but change the **pb.setting** configuration:

```
enforcehighsecurity no
ssl yes
ssloptions requiressl sslfirst sslverbose
sslservercertfile /etc/pbmasterhost.crt
sslserverkeyfile /etc/pbmasterhost.pem
networkencryption aes-256:keyfile=/etc/pbfips.key
iologencryption aes-256:keyfile=/etc/pbfips.key des:keyfile=/etc/oldpb.key
eventlogencryption aes-256:keyfile=/etc/pbfips.key
des:keyfile=/etc/oldpb.key
```

High security mode is not enabled, allowing the installation to read deprecated logs. SSL is enabled, with the correct configuration to allow the installation to communicate with the policy servers. The **iolog** and event log encryption must have FIPS 140-2 compatible algorithm specified if new logs are to be written. However this can be left out if the sole purpose of the installation is to read older logs. Appended to the end of the **iolog** and event log encryption configuration are the details of the customers' existing encryption used when the logs were encrypted. Privilege Management for Unix and Linux selects the relevant algorithm when the logs are replayed.

Set the Encryption Algorithm and Key

Starting with v8.0, the default encryption method is AES-256. If the encryption setting is commented out in the settings file, AES-256 encryption is used. Prior to v8.0, the default encryption method was DES.

Two parameters to consider when selecting an encryption algorithm are the block and key sizes. The larger an algorithm's block size, the more efficient it is. If an algorithm has multiple key sizes, then the larger the key, the more secure the algorithm.

A key must be generated for the use of encryption. The key must be the same on all the machines that are using the encryption. If the key is changed, then all of the encrypted files are no longer readable. Likewise, if the encryption algorithm is changed, any encrypted files are no longer readable.

As computing power and mathematical and algorithmic knowledge improves, so must encryption standards improve to keep data secure. Starting with v8.5, to move forward with new encryption standards, and to give the customer better control over their encryption standards, new configuration has been added. These improvements are the first phase of enhancements designed to ultimately make Privilege Management for Unix and Linux compliant with new government encryption standards such as FIPS 140-2.

Starting with v8.5, the settings file has been enhanced to incorporate the new keyword **enforcehighsecurity**<yes/no>.

The **iolog** and **eventlog** files are encrypted by their writers and are decrypted by their readers. Policy files must be encrypted manually and are decrypted by their readers. The default is **none** (no files encrypted).



Note: The settings files must be encrypted manually. An unencrypted copy of the settings file should be kept offline because Privilege Management for Unix and Linux does not provide a decryption program for the settings file. The installation suite does not work correctly if the settings file is encrypted. The unencrypted file needs to be restored before performing an upgrade, or before running **pemakeremotetar** or **pbuninstall**.



For more information, please see the following:

- ["enforcehighsecurity" on page 195](#)
- ["pbencode" on page 420](#)
- ["pbkey" on page 426](#)

enforcehighsecurity

- **Version 8.0 and earlier:** **enforcehighsecurity** setting not available.
- **Version 8.5 and later:** **enforcehighsecurity** setting available.

This enforces the use of more secure configuration, including using SSL for communications, FIPS 140-2 compliant symmetric encryption algorithms, an enhanced Pseudo Random Number Generator, and the use of the enhanced **pb.key** format.



Note: Only encryption algorithms that are accredited by FIPS 140-2 can be used for network and file encryption (for example, AES-128, AES-192, AES-256 and tripleDES). All others are deprecated.

Once this has been enabled the following **pb.settings** need to be configured:

- **ssl yes**
- **ssloptions requiressl sslfirst sslverbose**

- **sslengine**
- **sslservercertfile /etc/pbssl.pem**
- **sslcountrycode US**
- **sslprovince AZ**
- **ssllocality Phoenix**
- **sslorgunit Security**
- **sslorganization BeyondTrust**

You also need to generate a new key using **pbkey -F**.

**Example:**

```
enforcehighsecurity yes
```

Default

```
enforcehighsecurity yes
```

Used on

- Policy server hosts
- Submit hosts
- Run hosts



For more information, please see the following:

- ["pbkey" on page 426](#)
- ["ssl" on page 216](#)
- ["ssloptions" on page 217](#)
- ["sslservercertdir and sslservercertfile" on page 225](#)
- ["sslserverkeydir and sslserverkeyfile" on page 228](#)
- ["sslcountrycode" on page 197](#)
- ["sslprovince" on page 197](#)
- ["ssllocality" on page 198](#)
- ["sslorgunit" on page 199](#)
- ["sslorganization" on page 200](#)
- ["sslpruncipherlist" on page 221](#)
- ["sslservercipherlist" on page 226](#)

sslcountrycode

- **Version 8.5.0 and earlier:** `sslcountrycode` setting not available.
- **Version 9.0.0 and later:** `sslcountrycode` setting available.

Country code to use when creating x509 SSL client certificates. Used by Client Registration.



Example:

```
sslcountrycode US
```

Default

```
sslcountrycode US
```

Used on

All hosts



For more information, please see the following:

- ["ssl" on page 216](#)
- ["sslprovince" on page 197](#)
- ["ssllocality" on page 198](#)
- ["sslorgunit" on page 199](#)
- ["sslorganization" on page 200](#)
- ["sslprbruncipherlist" on page 221](#)
- ["sslservercipherlist" on page 226](#)

sslprovince

- **Version 8.5.0 and earlier:** `sslprovince` setting not available.
- **Version 9.0.0 and later:** `sslprovince` setting available.

Province to use when creating x509 SSL client certificates. Used by Client Registration.



Example:

```
sslprovince AZ
```

Default

```
sslprovince AZ
```

Used on

All hosts

 For more information, please see the following:

- ["ssl" on page 216](#)
- ["sslcountrycode" on page 197](#)
- ["ssllocality" on page 198](#)
- ["sslorgunit" on page 199](#)
- ["sslorganization" on page 200](#)
- ["sslpruncipherlist" on page 221](#)
- ["sslservercipherlist" on page 226](#)

ssllocality

- **Version 8.5.0 and earlier:** `ssllocality` setting not available.
- **Version 9.0.0 and later:** `ssllocality` setting available.

Locality to use when creating x509 SSL client certificates. Used by Client Registration.

 **Example:**


```
ssllocality Phoenix
```

Default

```
ssllocality Phoenix
```

Used on

All hosts

 For more information, please see the following:



- ["ssl" on page 216](#)
- ["sslcountrycode" on page 197](#)
- ["sslprovince" on page 197](#)
- ["sslorgunit" on page 199](#)
- ["sslorganization" on page 200](#)
- ["sslpruncipherlist" on page 221](#)
- ["sslservercipherlist" on page 226](#)

sslorgunit

- **Version 8.5.0 and earlier:** `sslorgunit` setting not available.
- **Version 9.0.0 and later:** `sslorgunit` setting available.

Organization unit to use when creating x509 SSL client certificates. Used by Client Registration.



Example:

```
sslorgunit Security
```

Default

```
sslorgunit Security
```

Used on

All hosts



For more information, please see the following:

- ["ssl" on page 216](#)
- ["sslcountrycode" on page 197](#)
- ["sslprovince" on page 197](#)
- ["ssllocality" on page 198](#)
- ["sslorganization" on page 200](#)
- ["sslpruncipherlist" on page 221](#)
- ["sslservercipherlist" on page 226](#)

sslorganization

- **Version 8.5.0 and earlier:** `sslorganization` setting not available.
- **Version 9.0.0 and later:** `sslorganization` setting available.

Organization to use when creating x509 SSL client certificates. Used by Client Registration.



Example:

```
sslorgunit BeyondTrust
```

Default

```
sslorgunit BeyondTrust
```

Used on

All hosts



For more information, please see the following:

- ["ssl" on page 216](#)
- ["sslcountrycode" on page 197](#)
- ["sslprovince" on page 197](#)
- ["ssllocality" on page 198](#)
- ["sslorgunit" on page 199](#)
- ["sslprbruncipherlist" on page 221](#)
- ["sslservercipherlist" on page 226](#)

networkencryption

- **Version 5.1 and earlier:** `networkencryption` setting not available.
- **Version 5.2 and later:** `networkencryption` setting available.

The `networkencryption` setting specifies one or more encryption settings for encrypting network traffic between hosts. The `networkencryption` setting uses the following syntax:

```
networkencryption <algorithm-1>:<keyfile=/fullpath/data-file-1>
[:<startdate=yyyy/mm/dd>:<enddate=yyyy/mm/dd>]
<algorithm-2>:<keyfile=/fullpath/data-file-2>
[:<startdate=yyyy/mm/dd>:<enddate=yyyy/mm/dd>] ...
```

where:

- **algorithm-n** is the name of the algorithm type.
- **/fullpath/data-file** (optional) specifies the full path and file name of the data file, which is used to dynamically derive the encryption key.
- **startdate=yyyy/mm/dd** specifies the earliest date that this algorithm is to be used.
- **enddate=yyyy/mm/dd** specifies the latest date that this algorithm is to be used.

Within each encryption setting, each component is separated by a colon (:). Multiple encryption settings are separated by a space.

For successful communications between Privilege Management hosts, each host must use the same encryption algorithm and data file, from which the encryption key is generated. To prevent service interruptions, you can specify multiple algorithms and keys on each host. The hosts resolve discrepancies as follows:

When one Privilege Management program attempts to communicate with another, it uses the first valid algorithm/key pair (encryption algorithm type and encryption key derived from the data file) in the **networkencryption** setting. The receiving host then attempts to find the correct algorithm/key pair from its **networkencryption** setting.

Servers attempt to connect the first valid algorithm/data-file pair and, if that fails, the servers then attempt to use other valid algorithm/data-file pairs that are defined in the **networkencryption** entry in the settings file. We strongly recommend placing the best and newest algorithm/data-file pair as the first entry in the settings file in all servers. Also, the algorithm/data-file pairs must be listed in the same order for all servers.

A client that is not upgraded to the newest algorithm/data-file pair continues to be supported by the policy server host as long as the client's algorithm/data-file pair is listed as a valid entry in the **networkencryption** setting. These clients continue to use the settings that are defined by the encryption keyword in the settings file, and the same initial algorithm/data-file pair is used during the initial connection between the two hosts.

If an algorithm/data-file pair is deprecated in the policy server host and it is the first item in the clients' list of supported algorithm/data-file pairs, then the new clients recognize this change and respond by automatically updating their setting files and backing up the previous settings files. Then the new clients reconnect to the policy server host using an algorithm/data-file pair that is common to both the policy server host and the client. However, if an algorithm/data-file pair is deprecated in the policy server host and the encryption that is used by the policy server host is not supported by the client, then the client's list must be manually upgraded or the initial connection will fail.

The starting date and ending dates are optional and are applied as follows:

- If the optional dates are used, then the algorithm/data-file pair is valid only during the specified time period.
- If a starting date is specified, then the algorithm/data-file pair takes effect at the start of that day; otherwise, the algorithm/data-file pair is active immediately.
- If an ending date is specified, the algorithm/data-file pair becomes inactive at the end of that date; otherwise, the algorithm/data-file pair never expires. The starting and ending dates are determined using Universal Coordinated Time (UTC) to eliminate ambiguity when the machines involved are in different time zones.
- If the optional dates are used, then the algorithm/data-file pair is valid only during the specified time period.
- If a starting date is specified, then the algorithm/data-file pair takes effect at the start of that day; otherwise, the algorithm/data-file pair is active immediately.
- If an ending date is specified, the algorithm/data-file pair becomes inactive at the end of that date; otherwise, the algorithm/data-file pair never expires. The starting and ending dates are determined using Universal Coordinated Time (UTC) to eliminate ambiguity when the machines involved are in different time zones.


IMPORTANT!

*If the start and/or end date option is used, administrators must ensure that all hosts use the same validity period. Failure to do so will result in the hosts being unable to communicate with each other, or the hosts using other less desirable algorithm/data-file pairs that are common to both hosts, and the hosts must be synchronized. If all listed algorithms have expired (they have an end date and the end date has expired), then the default network algorithm (DES) is used unless one of the network encryptions is listed or the keyword **none** is specified with no end date.*

This keyword supersedes the older encryption, keyfile, and encrypt keywords. The older settings are converted to the new standard when an upgrade installation occurs.


Example:

```
networkencryption des:keyfile=/etc/pb.key:enddate=2008/05/31 aes-
256:keyfile=/etc/pb.key.aes
```

This example setting directs the new client to use the DES encryption algorithm with the data file **/etc/pb.key** until May 31, 2008 (UTC). After that date, the new client is to use the AES-256 encryption algorithm with the data file **/etc/pb.key.aes**.

Default

The default encryption algorithm type is AES-256 and the default data file is typically **/etc/pb.key**.

Used on

- Policy server hosts
- Submit hosts
- Run hosts
- Log hosts
- Log synchronization hosts

eventlogencryption

- **Version 5.1 and earlier:** **eventlogencryption** setting not available.
- **Version 5.2 and later:** **eventlogencryption** setting available.

The **eventlogencryption** setting specifies one or more encryption settings for encrypting event logs. The **eventlogencryption** setting uses the following syntax:

```
eventencryption <algorithm-1>:<keyfile=/fullpath/data-file-1>
[:<startdate=yyyy/mm/dd>:<enddate=yyyy/mm/dd>]
<algorithm-2>:<keyfile=/fullpath/data-file-2>
[:<startdate=yyyy/mm/dd>:<enddate=yyyy/mm/dd>] ...
```

where:

- **algorithm-n** is the name of the algorithm type.
- **/fullpath/data-file** (optional) specifies the full path and file name of the data file, which is used to dynamically derive the encryption key.
- **startdate=yyyy/mm/dd** specifies the earliest date that this algorithm is to be used.
- **enddate=yyyy/mm/dd** specifies the latest date that this algorithm is to be used.

Within each encryption setting, each component is separated by a colon (:). Multiple encryption settings are separated by a space.



IMPORTANT!

When using multiple log servers, all log servers must use the same encryption algorithm and key. Otherwise, they cannot communicate with each other.

When a Privilege Management program attempts to write to an event log, it checks to determine if the event log uses the same algorithm/key pair (encryption algorithm and encryption key derived from the data file) as when the event log was created. If so, the event is written to the event log; otherwise, the old event log is archived and a new event log is started using the first available algorithm/key pair in the **eventlogencryption** setting. Algorithm/key pairs that are not active can still be used to read existing files.

The starting date and ending dates are optional and are applied as follows:

- If the optional dates are used, then the algorithm/data-file pair is valid only for writing to files during the specified time period.
- If a starting date is specified, then the algorithm/data-file pair takes effect at the start of that day; otherwise, the algorithm/data-file pair is active immediately.
- If an ending date is specified, then the algorithm/data-file pair becomes inactive at the end of that date, otherwise, the algorithm/data-file pair never expires. The starting and ending dates are determined using Universal Coordinated Time (UTC) to eliminate ambiguity when the machines involved are in different time zones.



Note: This keyword supersedes the older *encryption*, *keyfile*, and *encrypts keywords*. The older settings are converted to the new standard when an upgrade installation occurs.



Example:

```
eventlogencryption aes-256:keyfile=/etc/pb.key
```

*This example uses AES-256 encryption algorithm type with the encryption data file that is located in **/etc/pb.key**.*

Default

The default is no encryption.

Used on

Log hosts

iologencryption

- **Version 5.1 and earlier:** **iologencryption** setting not available.
- **Version 5.2 and later:** **iologencryption** setting available.

The **iologencryption** setting specifies one or more encryption settings for encrypting I/O logs. The **iologencryption** setting uses the following syntax:

```
iologencryption <algorithm-1>:<keyfile=/fullpath/data-file-1>
  [:<startdate=yyyy/mm/dd>:<enddate=yyyy/mm/dd>]
  <algorithm-2>:<keyfile=/fullpath/data-file-2>
  [:<startdate=yyyy/mm/dd>:<enddate=yyyy/mm/dd>] ...
```

where:

- **algorithm-n** is the name of the algorithm type.
- **/fullpath/data-file** (optional) specifies the full path and file name of the data file, which is used to dynamically derive the encryption key.
- **startdate=yyyy/mm/dd** specifies the earliest date that this algorithm is to be used.
- **enddate=yyyy/mm/dd** specifies the latest date that this algorithm is to be used.

Within each encryption setting, each component is separated by a colon (:). Multiple encryption settings are separated by a space.



IMPORTANT!

Caution! When using multiple log servers, all log servers must use the same encryption algorithm and key. Otherwise, they cannot communicate with each other, and **pbsync** cannot merge partial I/O logs.

When a Privilege Management program tries to write to an I/O log, it uses the first valid algorithm/key pair (encryption algorithm type and encryption key derived from the data file) in the **iologencryption** setting.

However, if the end date is reached and a session is still in place, the same algorithm/key pair is used until the end of the I/O log. If a session is abruptly terminated, then any new partial I/O log that is started will use a new algorithm/key pair. Algorithm/key pairs that are not active can still be used to read existing files.

The starting date and ending dates are optional and are applied as follows:

- If the optional dates are used, then the algorithm/data-file pair is only valid for writing to files during the specified time period.
- If a starting date is specified, then the algorithm/data-file pair takes effect at the start of that day; otherwise, the algorithm/data-file pair is active immediately.
- If an ending date is specified, then the algorithm becomes inactive at the end of that date, otherwise, the algorithm/data-file pair never expires.

The starting and ending dates are determined using Universal Coordinated Time (UTC). This eliminates ambiguity when the machines involved are in different time zones.



Note: This keyword supersedes the older encryption, keyfile, and encrypts keywords. The older settings are converted to the new standard when an upgrade installation occurs.

When a Privilege Management program attempts to read an I/O log, it searches the list of algorithm/key pairs to find the one that corresponds to the target file.

**Example:**

```
iologencryption des:keyfile=/etc/pb.key/des
```

Default

The default is no encryption.

Used on

Log hosts

reportencryption

- **Version 5.1 and earlier:** **reportencryption** setting not available.
- **Version 5.2 and later:** **reportencryption** setting available.

The **reportencryption** setting specifies one or more encryption settings for encrypting event log report control files. Each encryption setting consists of the encryption algorithm name, optional key file, optional starting date, and optional ending date using the following syntax:

```
reportencryption <algorithm-1>:<keyfile=/fullpath/data-file-1>  
[:<startdate=yyyy/mm/dd>:<enddate=yyyy/mm/dd>]  
<algorithm-2>:<keyfile=/fullpath/data-file-2>  
[:<startdate=yyyy/mm/dd>:<enddate=yyyy/mm/dd>] ...
```

where:

- **algorithm-n** is the name of the algorithm type.
- **/fullpath/data-file** (optional) specifies the full path and file name of the data file, which is used to dynamically derive the encryption key.
- **startdate=yyyy/mm/dd** specifies the earliest date that this algorithm is to be used.
- **enddate=yyyy/mm/dd** specifies the latest date that this algorithm is to be used.

Within each encryption setting, each component is separated by a colon (:). Multiple encryption settings are separated by a space.

When **pbguid** tries to write to a report control file it uses the first valid algorithm/key pair (encryption algorithm type and encryption key derived from the data file) in the **reportencryption** setting. When a Privilege Management for Unix and Linux program attempts to read a report control file it searches the list of algorithm/key pairs to find the one corresponding to the target file.

A list of algorithm/key pairs can be provided, but only the first valid entry is used for encryption purposes; all other entries are used as historical references to decrypt the report file. Algorithm/key pairs that are not active can still be used to read existing files.

The starting date and ending dates are optional and are applied as follows:

- If the optional dates are used, then the algorithm/data-file pair is only valid for writing to files during the specified time period.
- If a starting date is specified, then the algorithm/key data-file takes effect at the start of that day; otherwise, the algorithm/key data-file is active immediately.
- If an ending date is specified, then the algorithm becomes inactive at the end of that date; otherwise, the algorithm/key data-file never expires.

The starting and ending dates are reckoned using Universal Coordinated Time (UTC). Doing so eliminates ambiguity when the machines are in different time zones.



Note: This keyword supersedes the older `encryption`, `keyfile`, and `encrypts` keywords. The older settings are converted to the new standard when an upgrade installation occurs.



Example:

```
reportencryption saferplus-32:keyfile=/etc/pb.key.sp32
```

Default

The default is no encryption.

Used on

Log hosts

policyencryption

- **Version 5.1 and earlier:** `policyencryption` setting not available.
- **Version 5.2 and later:** `policyencryption` setting available.

The `policyencryption` setting specifies one or more encryption settings for encrypting policy files. Each encryption setting consists of the encryption algorithm name, optional key file, optional starting date, and optional ending date using the following syntax:

```
policyencryption <algorithm>:<keyfile=/fullpath/data-file>
```

where:

- **algorithm** is the name of the algorithm type.
- **/fullpath/data-file** (optional) specifies the full path and file name of the data file, which is used to dynamically derive the encryption key.

When `pbencode` or `pbguid` attempts to write to a policy file, it uses the algorithm/key pair (encryption algorithm type and encryption key derived from the data file) in the `policyencryption` setting. When a Privilege Management program attempts to read a policy file, it also uses the algorithm/key pair in the `policyencryption` setting.



Note: This keyword supersedes the older encryption, keyfile, and encrypts keywords. The older settings are converted to the new standard when an upgrade installation occurs.



Example:

```
policyencryption tripledes:keyfile=/etc/pb.key.3des
```

We recommend that you keep an unencrypted, offline copy of the policy file in the event you need to manually modify the file.

Default

The default is no encryption.

Used on

Policy server hosts

settingsencryptiontype

- **Version 5.1 and earlier:** `settingsencryptiontype` setting not available.
- **Version 5.2 and later:** `settingsencryptiontype` setting available.

The `pb.settings` file can be encrypted using the encryption algorithm type that is specified in the `settingsencryptiontype` keyword in the `pb.settings` file, using the following syntax:

```
settingsencryptiontype <algorithm>
```

The settings file is encrypted and decrypted using the default encryption key which is derived from the data file that is located in `/etc/<prefix>pb<suffix>.key`, and the encryption algorithm type that is defined in the `settingsencryption` keyword.



IMPORTANT!

We recommend you keep an unencrypted copy of the settings file in a secure location.



Note: This keyword supersedes the older encryption, keyfile, and encrypts keywords. The older settings are converted to the new standard when an upgrade installation occurs.


Example:

```
settingsencryptiontype des
```

In this example, DES is the encryption algorithm type to be used to encrypt the **pb.settings** file.

Default

The default is no encryption.

Used on

- Policy server hosts
- Submit hosts
- Run hosts
- Log hosts

dbencryption

- **Version 8.5 and earlier:** **dbencryption** setting not available.
- **Version 9.0 and later:** **dbencryption** setting available.

The **dbencryption** setting specifies the encryption for databases created by Privilege Management for Unix and Linux (e.g. **clntregdb**, **eventdb**, **svccachedb**, **logarchivedb**, et. al).

The **dbencryption** setting uses the following syntax:

```
dbencryption <algorithm-1>:<keyfile=/fullpath/data-file-1>
[:<startdate=yyyy/mm/dd>:<enddate=yyyy/mm/dd>]
<algorithm-2>:<keyfile=/fullpath/data-file-2>[:<startdate=yyyy/mm/dd>:<enddate=yyyy/mm/dd>] ...
```

where:

- **algorithm-n** is the name of the algorithm type.
- **/fullpath/data-file** (optional) specifies the full path and file name of the data file, which is used to dynamically derive the encryption key.
- **startdate=yyyy/mm/dd** specifies the earliest date that this algorithm is to be used.
- **enddate=yyyy/mm/dd** specifies the latest date that this algorithm is to be used.

Within each encryption setting, each component is separated by a colon (:). Multiple encryption settings are separated by a space.

Default

none

Used On

All hosts

Pluggable Authentication Modules

Privilege Management for Unix and Linux can use Pluggable Authentication Modules (PAM) on systems where it is available. Privilege Management for Unix and Linux invokes password authentication services, account management services, and session start/end services.

pam

- **Version 3.5 and earlier:** pam setting not available.
- **Version 4.0 and later:** pam setting available.

The **pam** setting enables the use of PAM if set to **yes**, or disables it if set to **no**.



Example:

```
pam yes
```

Default

```
pam no
```

Used on

- Policy server hosts
- Submit hosts
- Run hosts

libpam

- **Version 5.1.1 and earlier:** libpam setting not available.
- **Version 5.2 and later:** libpam setting available.

libpam is a user-defined PAM library that Privilege Management for Unix and Linux uses as a first option in case the system does not use the standard default PAM libraries. The notation used for AIX to specify the OS-provided PAM library is the following:

```
/usr/lib/libpam.a(shr.o)
```



Example:

```
libpam /lib/libpam.so.1
```

Default

No default value

Used on

- Policy server hosts
- Submit hosts
- Run hosts

pampasswordservice

- **Version 3.5 and earlier:** `pampasswordservice` setting not available.
- **Version 4.0 and later:** `pampasswordservice` setting available.

If you want Privilege Management for Unix and Linux to use PAM password authentication and account management for password authentication, set `pampasswordservice` to the name of the PAM service that you want to use.

- On a policy server host, PAM password authentication is used for the `getuserpasswd()` function.
- On a submit host, PAM password authentication is used when the `submitconfirmuser()` function is invoked by the policy server host's policy.
- On a run host, PAM password authentication is used when `runconfirmuser` is invoked by the policy server host's policy.



Note: Privilege Management for Unix and Linux does not use the environment variables that are set by `pam_env`. Privilege Management for Unix and Linux can read environment variables from `/etc/environment` or some other file. For more information, please see the following:

- "`environmentfile`" on page 72,
- "`runenvironmentfile`" in the [Privilege Management for Unix and Linux Policy Language Guide](https://www.beyondtrust.com/docs/privilege-management/unix-linux/index.htm) at <https://www.beyondtrust.com/docs/privilege-management/unix-linux/index.htm>.



Example:

```
pampasswordservice login
```

Default

No default value

Used on

- Policy server hosts
- Submit hosts

- Run hosts



Note: Many Privilege Management for Unix and Linux programs run as **root**. If you use a PAM service that allows root to bypass passwords (for example, **su** or anything containing **rootok**), then Privilege Management for Unix and Linux may also skip the password check.

pamsessionsservice

- **Version 3.5 and earlier:** **pamsessionsservice** setting not available.
- **Version 4.0 and later:** **pamsessionsservice** setting available.

If you want PAM to perform account management and session start and end services to manage task requests on a run host, then set **pamsessionsservice** to the name of the service that you want to use. **pblocald** invokes the account management and session start portions when the requested task starts, and invokes session end services when the requested task finishes.

For local mode, the client invokes the account management module when the **runuser** is different than the submitting user (**user**). Unless I/O logging is active, session start and end services are skipped.

In version 6.0 and later, Privilege Management for Unix and Linux uses **ulimits** that are set by **pam_limits** during PAM session start. If you do not want to honor the **ulimits** that are set by PAM, use the **pam_session_prepb6** setting.



Note: Privilege Management for Unix and Linux does not use the environment variables that are set by **pam_env**.



Note: Privilege Management for Unix and Linux can read environment variables from **/etc/environment** or some other file.

For more information, please see the following:

- "**environmentfile**" on page 72
- "**runenvironmentfile**" in the [Privilege Management for Unix and Linux Policy Language Guide](https://www.beyondtrust.com/docs/privilege-management/unix-linux/index.htm) at <https://www.beyondtrust.com/docs/privilege-management/unix-linux/index.htm>



Example:

```
pamsessionsservice su
```

Default


No default value



Note: Some PAM services may update the **syslog** and the **utmp** or **utmpx** files. To avoid duplicate entries, you might need to set **recordunixptysessions** and **syslogsessions** to **no**.

Used on


- Run hosts
- Submit hosts by **pbksh** and **pbsh**


 For more information, please see "[pam_session_prepb6](#)" on page 213.

pamsuppresspbpasswordprompt

- **Version 5.1.1 and earlier:** `pamsuppresspbpasswordprompt` setting not available.
- **Version 5.1.2 and later:** `pamsuppresspbpasswordprompt` setting available.

If you want to suppress the Privilege Management for Unix and Linux password prompt when PAM authentication is enabled, then set `pamsuppresspbpasswordprompt` to `yes`. Otherwise, if the Privilege Management for Unix and Linux password prompt is required, then set `pamsuppresspbpasswordprompt` to `no`.

 **Note:** If the values of the `user` and `runuser` variables are different, the Privilege Management for Unix and Linux password prompt is always enabled, even if `pamsuppresspbpasswordprompt` is set to `yes`.

 **Example:**

```
pamsuppresspbpasswordprompt yes
```

Default

```
pamsuppresspbpasswordprompt yes
```

Used on

- Policy server hosts
- Submit hosts
- Run hosts

pam_session_prepb6

- **Version 5.2 and earlier:** `pam_session_prepb6` setting not available.
- **Version 6.0 and later:** `pam_session_prepb6` setting available.

Prior to Privilege Management for Unix and Linux version 6, the PAM session is called by the parent Privilege Management for Unix and Linux process. In version 6, this behavior was corrected so that the PAM session is called from the child process that runs the secured task. By setting **pam_session_prepb6** to **yes**, you can revert Privilege Management for Unix and Linux to the old behavior.

**Example:**

```
pam_session_prepb6 yes
```

Default

```
pam_session_prepb6 no
```

Used on

Run hosts

pamsetcred

- **Version 6.0 and earlier:** **pamsetcred** setting not available.
- **Version 6.1 and later:** **pamsetcred** setting available.

The **pamsetcred** keyword enables the **pam_setcred()** function, which is used to establish possible additional credentials of a user.



Note: This keyword does not apply to **pbssh**. If it is present in the settings file, it does not have any effect on **pbssh** and is ignored.

For Solaris projects, this sets the Project ID to the default project, or to a specified project. Other scenarios are possible, depending on the OS PAM implementation and configuration.



Note: The use of **pam_setcred** currently does not delete credentials after a session.

**Example:**

```
pamsetcred yes
```

Default

```
pamsetcred no
```

Used on

Run hosts

Secure Socket Layers and Public Key Infrastructure

Secure Socket Layers (SSL) enables the use of digital certificates, certificate authorities, extensive network encryption, and checksums for all network packets.

Starting with v3.0, Privilege Management supports Public Key Infrastructure (PKI) through SSL. This feature enables the use of Privacy Enhanced Mail (PEM) format certificates, private keys, and certificate authority files. The SSL features are controlled through the **ssl** setting, the **ssloptions** setting, and the client and server settings.

Many of the SSL settings enable token expansion for some useful strings. These are summarized in the following table.

SSL Parameter Substitutions

Symbol	Replacement
%%	A % character.
%g	User's group ID.
%G	User's group ID number.
%h	Local host name. The unqualified name of the current machine.
%H	Remote host name of the current machine in Fully Qualified Domain Name (FQDN) format (if available from uname).
%l	Unqualified local host name as determined by the network interface.
%L	Local host interface name. The local host name, as determined by the network interface, in FQDN format (if available).
%n	Program name with neither a prefix or suffix.
%N	Program name with a prefix and suffix.
%p	Program prefix.
%r	Unqualified host name, as determined by the network interface.
%R	Remote host interface name. The remote host name, as determined by the network interface, in FQDN format.
%s	Program suffix.
%u	User's login ID.
%U	User's UID.

ssl

When set to **yes**, the **ssl** setting enables the use of Privilege Management for Unix and Linux SSL features. When set to **no**, the **ssl** setting disables the use of these features.


Example:

```
ssl yes
```

Default

```
ssl yes
```

Used on

- Log hosts
- Policy server hosts
- Submit hosts
- Run hosts

restssloptions

- **Version 10.1.0 and earlier:** `restssloptions` setting not available.
- **Version 10.2.0 and later:** `restssloptions` setting available.

The current `restssloptions` include:

- **TLSTMinV1**, **TLSTMinV1.0**, **TLSTMinV1.1**, **TLSTMinV1.2**, and **TLSTMinV1.3**
- **TLSTMaxV1**, **TLSTMaxV1.0**, **TLSTMaxV1.1**, **TLSTMaxV1.2** and **TLSTMaxV1.3**
- **MinHMACMD5** and **MinHMACSHA512**

For FIPS compliance, all Privilege Management for Unix and Linux hosts must add **MinHMACSHA512** to the `restssloptions` setting.

ssloptions

- **Version 4.0.0 and later:** `ssloptions` setting available.

The `ssloptions` setting controls the following system-wide options:

Option	Description
ClientCertificates	To require certificates on the client side, add ClientCertificates to the <code>ssloptions</code> line.
AllowNonSSL	To communicate with older, non-SSL versions of Privilege Management for Unix and Linux, add AllowNonSSL to your <code>ssloptions</code> line. Doing so allows SSL-enabled versions to communicate with non-SSL versions. If a Privilege Management for Unix and Linux client is SSL-enabled and the policy server host specifies AllowNonSSL , but not ClientCertificates , then the communications do not use SSL.

Option	Description
TLSMinV1.0, TLSMinV1.1, TLSMinV1.2, TLSMinV1.3	When SSL is enabled, this option allows you to set the minimum SSL/TLS value to use in the protocol.
TLSMaxV1.0, TLSMaxV1.1, TLSMaxV1.2, TLSMaxV1.3	When SSL is enabled, this option allows you to set the maximum SSL/TLS value to use in the protocol.
RequireSSL	<p>To require SSL communications between Privilege Management components without requiring Privilege Management for Unix and Linux client certificates, then add RequireSSL to your ssloptions line.</p> <p>This option is not compatible with the AllowNonSSL option. If you specify both AllowNonSSL and RequireSSL, then the last one that is specified takes precedence.</p>
SSLFirst	<p>If the SSLFirst option is selected, this option forces the SSL handshake to happen before the Privilege Management for Unix and Linux handshake.</p> <p>The SSLFirst option must be set on every Privilege Management for Unix and Linux host including clients and servers.</p> <p>The SSLFirst option is turned on by default in version 10.3.2 and later.</p>
sslverbose	If the sslverbose option is selected, server components log informational messages that are sent to error logs, detailing connections, SSL/TLS protocols, and the encryption ciphers used to communicate. This is a debugging and diagnostic option
validateClient	<p>The option validateClient enables Privilege Management for Unix and Linux servers (pbmasterd, pblocald, pblogd) to use SSL verifypeer and verifyhost features to validate the connected client host. Note that pbmasterd is also a client to pblocald, and both pbmasterd and pblocald are clients to pblogd.</p> <p>This can be used when the client hosts have certificates installed, and the servers' ssloptions includes the ClientCertificates option (validateClient forces ClientCertificates).</p> <p>Enabling the validateClient ssloption on the server requires that pb.settings on the server includes the sslservercafile keyword, specifying the CA that signed the client's certificate. The pb.settings file on the client must include the sslpbbruncertfile and sslpbbrunkeyfile keywords, specifying the client's certificate and key. This feature alternatively uses the sslpbbruncertdir, sslpbbrunkeydir, and sslservercadir keywords.</p> <p>The pb.settings file on pbmasterd and pblocald must include sslservercertfile and sslserverkeyfile keywords, specifying the client's certificate and key. This feature alternatively uses the sslservercertdir and sslserverkeydir keywords.</p> <p>Enabling the AllowNonSSL with validateClient results in an error. Non-SSL connections are not allowed with validateClient.</p> <p>The client host's hostname should be listed in the Subject Alternative Name (SAN) field of the certificate.</p>
validateServer	<p>The option validateServer enables Privilege Management for Unix and Linux SSL clients to verify the server with the SSL verifypeer and verifyhost features. Note that pbmasterd is a client to pblocald, and both pbmasterd and pblocald are clients to pblogd.</p> <p>Enabling the validateServer on the client requires that pb.settings on the client includes the sslpbbruncafile keyword (sslpbservercafile keyword on pbmasterd and pblocald), specifying the CA that signed the server's certificate. The pb.settings file on the server must include the</p>

Option	Description
	<p>sslservercertfile and sslserverkeyfile keywords, specifying the server's certificate and key. This feature alternatively uses the sslservercertdir, sslserverkeydir, and sslpbruncadir keywords.</p> <p>Enabling the AllowNonSSL with validateServer results in an error. Non-SSL connections are not allowed with validateServer.</p> <p>The hostname should be listed in the Subject Alternative Name (SAN) field of the certificate.</p>



Note: The program terminates if invalid values are provided for **ssloptions**.



Example:

```
ssloptions AllowNonSSL
ssloptions requiressl sslfirst
ssloptions ClientCertificates
ssloptions AllowNonSSL ClientCertificates
```

Default

```
requiressl
```

Used on

- Log hosts
- Policy server hosts
- Submit hosts
- Run hosts

Server-Side SSL

For client hosts where optimized run mode is always used (the submit host is always the run host), a server-side SSL scenario can be set up where the client machine does not need a server key/certificate pair or a client key/certificate pair.

The **sslpbruncafile** keyword is optional. If **sslpbruncafile** is specified, **sslpbruncafile** is the certificate authority (CA) that signed the server's certificate. If **sslpbruncafile** is not specified, then the server's certificate authenticity is not verified.



Note: If the submit host is not the same host as the run host or if a log server is not used, then the **pblocald** server is used to execute the secured task. **pblocald** is an SSL server and requires the **sslservercafile**, **sslservercertfile**, and **sslserverkeyfile** settings.

SSL Client Settings

The SSL client settings configure SSL for Privilege Management for Unix and Linux client programs.

sslpruncafile and sslpruncadir

- **Version 4.0.0 and later:** `sslpruncafile` and `sslpruncadir` settings available.

These settings specify the path to a certificate authority directory or file.

A certificate authority file is a PEM-formatted file that contains one or more PEM-formatted signature certificates. The programs `pbrun`, `pbksh`, and `pbsh` use these certificate authority files to validate certificates from `pbmasterd` and `pblocald`. This file should not contain private keys.

If `sslpruncafile` contains an absolute path, then that file is used as the certificate authority file. If `sslpruncafile` contains a relative path, then the value of the `sslpruncadir` setting is prepended to form an absolute path. The `pbrun` certificate authority file and certificate authority directory must be owned by root and no one else should have write permission.



These settings enable the parameter substitutions shown in "SSL Parameter Substitutions" on page 216.



Example:

```
sslpruncafile /secure/ca/pbrun/OurAuthority.pem
sslpruncadir /secure/ca/pbrun
sslpruncafile OurAuthority.pem
sslpruncadir /secure/ca/pbrun sslpruncafile %N.pem
```

Default

No default value

Used on

Submit hosts

sslpruncertdir and sslpruncertfile

- **Version 4.0.0 and later:** `sslpruncertdir` and `sslpruncertfile` settings available.

The `sslpruncertdir` and `sslpruncertfile` settings specify the path of a Privacy Enhanced Mail (PEM) format certificate file for clients to communicate with `pbmasterd` and `pblocald`.

If a full absolute path is provided for `sslpruncertfile`, then it is used. If a relative path is provided for `sslpruncertfile`, then the directory specified in the `sslpruncertdir` setting is prepended to form the certificate file path.

root or the submitting user must own the `pbrun` certificate file and certificate directory. No one else should have write permission.

i These settings allow the parameter substitutions that are shown in "SSL Parameter Substitutions" on page 216.

o *Example:*

```

sslpbruncertfile /secure/certificates/pbrun/pbrun.pem
sslpbruncertdir /secure/certificates/pbrun
sslpbruncertfile pbrun.pem
sslpbruncertdir /home/%u/certificates
sslpbruncertfile %u.pem
    
```

Defaults

No default value

Used on

Submit hosts

sslpbruncipherlist

- **Version 4.0.0 and later:** **sslpbruncipherlist** setting available.

OpenSSL provides a variety of algorithms that can be used for encryption. The **sslpbruncipherlist** setting enables the administrator to restrict or promote the set of encryption algorithms that are used by Privilege Management clients to communicate with SSL enabled server services.

The keyword **sslpbruncipherlist** accepts "**cipherlist=**" and "**tlsv1.3=**" cipher groups.

The cipher groups **cipherlist=** and **tlsv1.3=** are case-sensitive and space is not allowed before =.

When using the **sslpbruncipherlist** keyword, the order of cipher lists is not relevant.

This format: **sslpbruncipherlist cipherlist= TLSv1.2:!SSLv2:@STRENGTH tlsv1.3= TLS_AES_256_GCM_SHA384**

is the same as this format:

sslpbruncipherlist tlsv1.3= TLS_AES_256_GCM_SHA384 cipherlist= TLSv1.2:!SSLv2:@STRENGTH

These ciphers are limited to the set of ciphers available in the given version of OpenSSL used by the Privilege Management installation.

i For more information, please see the [Release Notes](https://www.beyondtrust.com/docs/release-notes) at <https://www.beyondtrust.com/docs/release-notes>.

Valid Values

Refer to the following table for the valid values for the **sslpbruncipherlist**. To use more than one cipher set, separate the values with colons.

cipherlist Values

OpenSSL Cipher Set	Setting Value
SSL_RSA_WITH_NULL_MD5	NULL-MD5
SSL_RSA_WITH_NULL_SHA	NULL-SHA
SSL_RSA_EXPORT_WITH_RC4_40_MD5	EXP-RC4-MD5
SSL_RSA_WITH_RC4_128_MD5	RC4-MD5
SSL_RSA_WITH_RC4_128_SHA	RC4-SHA
SSL_RSA_EXPORT_WITH_RC2_CBC_40_MD5	EXP-RC2-CBC-MD5
SSL_RSA_EXPORT_WITH_DES40_CBC_SHA	EXP-DES-CBC-SHA
SSL_RSA_WITH_DES_CBC_SHA	DES-CBC3-SHA
SSL_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA	EXP-EDH-DSS-DES-CBC-SHA
SSL_DHE_DSS_WITH_DES_CBC_SHA	EDH-DSS-CBC-SHA
SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA	EDH-DSS-DES-CBC3-SHA
SSL_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA	EXP-EDH-RSA-DES-CBC-SHA
SSL_DHE_RSA_WITH_DES_CBC_SHA	EDH-RSA-DES-CBC-SHA
SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA	EDH-RSA-DES-CBC3-SHA

tls1.3 Values

OpenSSL Cipher Set	Setting Value
TLS13-AES-256-GCM-SHA384	TLS13-AES-256-GCM-SHA384
TLS13-CHACHA20-POLY1305-SHA256	TLS13-CHACHA20-POLY1305-SHA256
TLS13-AES-128-GCM-SHA256	TLS13-AES-128-GCM-SHA256
TLS13-AES-128-CCM-8-SHA256	TLS13-AES-128-CCM-8-SHA256
TLS13-AES-128-CCM-SHA256	TLS13-AES-128-CCM-SHA256

Examples

In the following code snippet, PMUL uses the cipher lists:

- **TLSv1.2:!SSLv2:@STRENGTH** for TLS v1.2 (and earlier) connections
- **TLS_AES_256_GCM_SHA384:TLS_AES_128_GCM_SHA256** for TLS v1.3 connections


Example:

```
sslpruncipherlist cipherlist=TLSv1.2:!SSLv2:@STRENGTH tlsv1.3=TLS_AES_256_GCM_
SHA384:TLS_AES_128_GCM_SHA256
```

In the following code snippet, PMUL uses the cipher lists:

- **TLSv1.2:!SSLv2:!3DES:!eNULL:@STRENGTH** for TLS v1.2 (and earlier) connections.
- **tlsv1.3=** cipher group for TLSv1.3 connections. This is the default value.


Example:

```
sslpruncipherlist cipherlist=TLSv1.2:!SSLv2:!3DES:!eNULL:@STRENGTH
```

Default

```
cipherlist=TLSv1.2:!SSLv2:!3DES:!MD5:!ADH:!AECDH:!DHE:!eNULL:@STRENGTH
```

```
tlsv1.3=TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_GCM_SHA256
```

Used on

Submit hosts

sslprunkeydir and sslprunkeyfile

- **Version 4.0.0 and later:** **sslprunkeydir** and **sslprunkeyfile** settings available.

The **sslprunkeyfile** and **sslprunkeydir** settings enable you to specify the location of a PEM-formatted private key for the client certificate file that is used to communicate with **pbmasterd** and **pblocald**.

If **sslprunkeyfile** is a full path name, then it is used for the private key. If **sslprunkeyfile** does not contain an absolute path, then **sslprunkeydir** is prepended to it.

The clients are usually interactive, so the private keys can be encrypted. The clients prompt for the passphrase when needed. If you are invoking a client non-interactively (for example, from **cron**), then the private key should not be encrypted.

root or the submitting user must own the private key file and the private key directory. No one else should have read or write permission.

If the key file and directory are not set, then the client looks in the certificate file to see if the key is there. In this case, the certificate file and directory must be read-only and owned by root or the submitting user. No one else should have read or write permission.



Note: These settings enable the parameter substitutions that are shown in "[SSL Parameter Substitutions](#)" on page 216.

**Example:**

```
sslpbrunkeyfile /secure/privatekeys/pbrun.pem
sslpbrunkeydir /secure/privatekeys/
sslpbrunkeyfile %u.pem
sslpbrunkeydir /home/%u/privatekeys
sslpbrunkeyfile %u.pem
```

Defaults

No default value

Used on

Submit hosts

sslpbrunverifysubject

- **Version 4.0.0 and later:** `sslpbrunverifysubject` setting available.

`sslpbrunverifysubject` contains a series of regular expressions to check against the policy server's certificate subject line. If the subject line matches all patterns, then the connection is allowed to proceed. If any of the patterns do not match, then the connection fails.



Example: This example verifies that the **CN** attribute (common name) matches the host name of the remote machine:

```
sslpbrunverifysubject /CN=%R/
```



Example: This example verifies that the **O** attribute equals **Company Name** and that the **OU** attribute starts with **Technology**:

```
sslpbrunverifysubject '/O=Company Name/' /OU=Technology
```



Note: Single quotation marks should surround the attribute if there are embedded spaces.

Default

No default value

Used on

Submit hosts

SSL Server Settings

The SSL server settings configure SSL for Privilege Management for Unix and Linux server programs.

sslservercadir and sslservercafile

- **Version 4.0.0 and later:** `sslservercadir` and `sslservercafile` settings available.

The `sslservercadir` and `sslservercafile` settings specify the path to a certificate authority directory or file. A certificate authority file is a PEM-formatted file that contains one or more PEM-formatted signature certificates that are used to validate server certificates. This file should not contain private keys.

If `sslservercafile` contains an absolute path, then that file is used as the certificate authority file. If `sslservercafile` contains a relative path, then the value of the `sslservercadir` setting is prepended to form an absolute path.

The server certificate authority file and certificate authority directory must be owned by root and no one else should have write permission.



Note: These settings allow the parameter substitutions that are shown in "[SSL Parameter Substitutions](#)" on page 216.



Example:

```
sslservercafile /secure/ca/servers/OurAuthority.pem
sslservercadir /secure/ca/servers
sslservercafile OurAuthority.pem
sslservercadir /secure/ca/servers sslservercafile %h.pem
```

Defaults

```
/etc/<prefix>pbssl.pem<suffix>
```

Used on

- GUI hosts
- Log hosts
- Policy server hosts
- Run hosts

sslservercertdir and sslservercertfile

- **Version 4.0.0 and later:** `sslservercertdir` and `sslservercertfile` settings available.

The **sslservercertdir** and **sslservercertfile** settings specify the path of a Privacy Enhanced Mail (PEM) format certificate file for **pbmasterd**, **pblocald**, **pblogd**, and **pbguid** to communicate with each other or with client programs. If a full absolute path is provided for **sslservercertfile**, then it is used as specified. If a relative path is provided for **sslservercertfile**, then the directory specified in the **sslservercertdir** setting is prepended to form the certificate file path.

The server certificate file and certificate directory must be owned by root and no one else should have write permission.



Note: These settings allow the parameter substitutions that are shown in "SSL Parameter Substitutions" on page 216.



Example:

```
sslservercertfile /secure/certificates/servers/pbmasterd.pem
sslservercertdir /secure/certificates/servers
sslservercertfile pbmasterd.pem
sslservercertdir /secure/certificates/servers
sslservercertfile %N.pem
```

Defaults

/etc/pbssl.pem

Used on

- GUI hosts
- Log hosts
- Policy server hosts
- Run hosts

sslservercipherlist

- **Version 4.0.0 and later:** **sslservercipherlist** setting available.

OpenSSL provides a variety of algorithms that can be used for encryption. The **sslservercipherlist** setting enables the administrator to restrict or promote the set of encryption algorithms that are used by Privilege Management servers when they receive communications from SSL enabled clients.

These ciphers are limited to the set of ciphers available in the given version of OpenSSL used by the Privilege Management installation.

The keyword **sslservercipherlist** accepts "**cipherlist=**" and "**tlsv1.3=**" cipher groups.

The cipher groups **cipherlist=** and **tlsv1.3=** are case-sensitive and space is not allowed before **=**.

When using the **sslservercipherlist** keyword, the order of cipher lists is not relevant.

This format: **sslservercipherlist cipherlist= TLSv1.2:!SSLv2:@STRENGTH tlsv1.3= TLS_AES_256_GCM_SHA384**

is the same as this format:

```
sslservercipherlist tlsv1.3= TLS_AES_256_GCM_SHA384 cipherlist= TLSv1.2:!SSLv2:@STRENGTH
```

Valid Values

To use more than one cipher set, separate the values with colons.

Examples



Example:

```
sslservercipherlist  
cipherlist=TLSv1.2:!SSLv2:!3DES:!MD5:!ADH:!AECDH:!DHE:!eNULL:@STRENGTH tlsv1.3=TLS_AES_  
256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_GCM_SHA256
```



Example:

```
sslservercipherlist  
cipherlist=TLSv1.2:!SSLv2:!3DES:!MD5:!ADH:!AECDH:!DHE:!eNULL:@STRENGTH
```



Example:

```
sslservercipherlist tlsv1.3=TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256:TLS_AES_  
128_GCM_SHA256:TLS_AES_128_CCM_SHA256:TLS_AES_128_CCM_8_SHA256
```

Default

Default cipherlist value for the cipher group cipherlist:

```
cipherlist=TLSv1.2:!SSLv2:!3DES:!MD5:!ADH:!AECDH:!DHE:!eNULL:@STRENGTH
```

Default cipher suite value of the cipher group tlsv1.3:

```
tlsv1.3=TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_GCM_SHA256
```

Used on

- GUI hosts
- Log hosts
- Policy server hosts
- Run hosts



For more information, please see the following:

- [Release Notes](https://www.beyondtrust.com/docs/release-notes) at <https://www.beyondtrust.com/docs/release-notes>
- "cipherlist Values" on page 222

sslserverkeydir and sslserverkeyfile

- **Version 4.0.0 and later:** `sslserverkeydir` and `sslserverkeyfile` settings available.

The `sslserverkeyfile` and `sslserverkeydir` settings enable you to specify the location of a PEM- formatted private key for the server certificate file that is used by `pbmasterd`, `pblocald`, `pblogd`, and `pbguid` to communicate with each other or with client programs.

If `sslserverkeyfile` is a full path name, then it is used for the private key. If `sslserverkeyfile` does not contain an absolute path, then `sslserverkeydir` is prepended to it.

The servers are not interactive, so the private keys should not be encrypted.

The private key file and the private key directory must be owned by `root` and no one else should have read or write permission.

If the key file and directory are not set, then the daemons look in the certificate file to see if the key is there. In this case, the certificate file and directory must be read-only and owned by `root`. No one else should have read or write permission.



Note: These settings allow the parameter substitutions that are shown in "SSL Parameter Substitutions" on page 216.



Example:

```
sslserverkeyfile /secure/certificates/serverkeys/pbmasterd.pem
sslserverkeydir /secure/certificates/serverkeys
sslserverkeyfile pbmasterd.pem
sslserverkeydir /secure/certificates/serverkeys
sslserverkeyfile %N.pem
```

Defaults

```
/etc/<prefix>pbssl.pem<suffix>
```

Used on

- GUI hosts
- Log hosts
- Policy server hosts
- Run hosts

sslsrvrverifysubject

- **Version 4.0.0 and later:** `sslsrvrverifysubject` setting available.

`sslsrvrverifysubject` contains a series of regular expressions to check against the client's or other server's certificates subject line. If the subject line matches all patterns, then the connection is allowed to proceed. If any of the patterns do not match, then the connection fails.



Example: This example verifies that the **CN** attribute (common name) matches the host name of the remote machine:

```
sslsrvrverifysubject /CN=%R/
```



Example: This example verifies that the **O** attribute equals **Company Name** and the **OU** attribute starts with **Technology**:

```
sslsrvrverifysubject '/O=Company Name/' /OU=Technology
```



Note: Single quotation marks should surround the attribute if there are embedded spaces.

Default

No default value

Used on

- GUI hosts
- Log hosts
- Policy server hosts
- Run hosts

Additional Configuration to Improve PMUL Security

Privilege Management for Unix and Linux does not contain a Certificate Authority (CA), therefore certificates generated during install are self-signed, and cannot be used to properly identify the host. Creating and deploying proper x509 certificates, with hostname information in the Subject Alternative Name field, allows Privilege Management for Unix and Linux hosts to properly identify hosts. TLS clients can verify the server's certificate and hostname by adding the `validateServer` option to the `ssloptions` keyword in `/etc/pb.settings`. For TLS, `pbmasterd` and `pblocald` are clients to `pblogd`. Additionally, servers can validate the certificates and hostnames of the client hosts by adding the `validateClient` option to the `ssloptions` keyword in `/etc/pb.settings`.

Configure Privilege Management for Unix and Linux to use the `SSLFirst` keyword in `/etc/pb.settings`. This keyword must have the same value on all hosts in the PMUL domain. The `SSLFirst` keyword results in SSL/TLS occurring prior to any Privilege Management for Unix and Linux proprietary protocol negotiations that use symmetric keys, reducing any issue with compromised symmetric network encryption keys.

The TLS ciphers should be changed to disallow anonymous ciphers.

Edit the **sslpruncipherlist** and **sslservercipherlist** entries in **/etc/pb.settings**:

```
sslpruncipherlist      cipherlist=TLSv1.2:!SSLv2:!3DES:!MD5:!ADH:!AECDH:!DHE:!eNULL:@STRENGTH
tlsv1.3=TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_GCM_SHA256
```

```
sslservercipherlist   cipherlist=TLSv1.2:!SSLv2:!3DES:!MD5:!ADH:!AECDH:!DHE:!eNULL:@STRENGTH
tlsv1.3=TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256:TLS_AES_128_GCM_SHA256
```

Edit the **ssl.cipher-list** entry in **/usr/lib/beyondtrust/pb/rest/etc/pbhttpd.conf**:

```
ssl.cipher-list       = "TLSv1.2:!SSLv2:!3DES:!MD5:!ADH:!AECDH:!DHE:!eNULL:@STRENGTH"
```

and

```
ssl.openssl.ssl-conf-cmd = (
    "MinProtocol" => " TLSv1.2",
    "CipherString" =>
"TLSv1.2:!SSLv2:!3DES:!MD5:!ADH:!AECDH:!DHE:!eNULL:@STRENGTH",
    "Ciphersuites" => "TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256:TLS_
AES_128_GCM_SHA256"
)
```



Note: PMUL version 21.1 and below of PMUL client registration uses TLSv1. Use below TLS protocol version to allow older versions of PMUL client registrations.

```
ssl.cipher-list = "HIGH:!SSLv2:!3DES:!MD5:!ADH:!AECDH:!DHE:!eNULL:@STRENGTH"
```

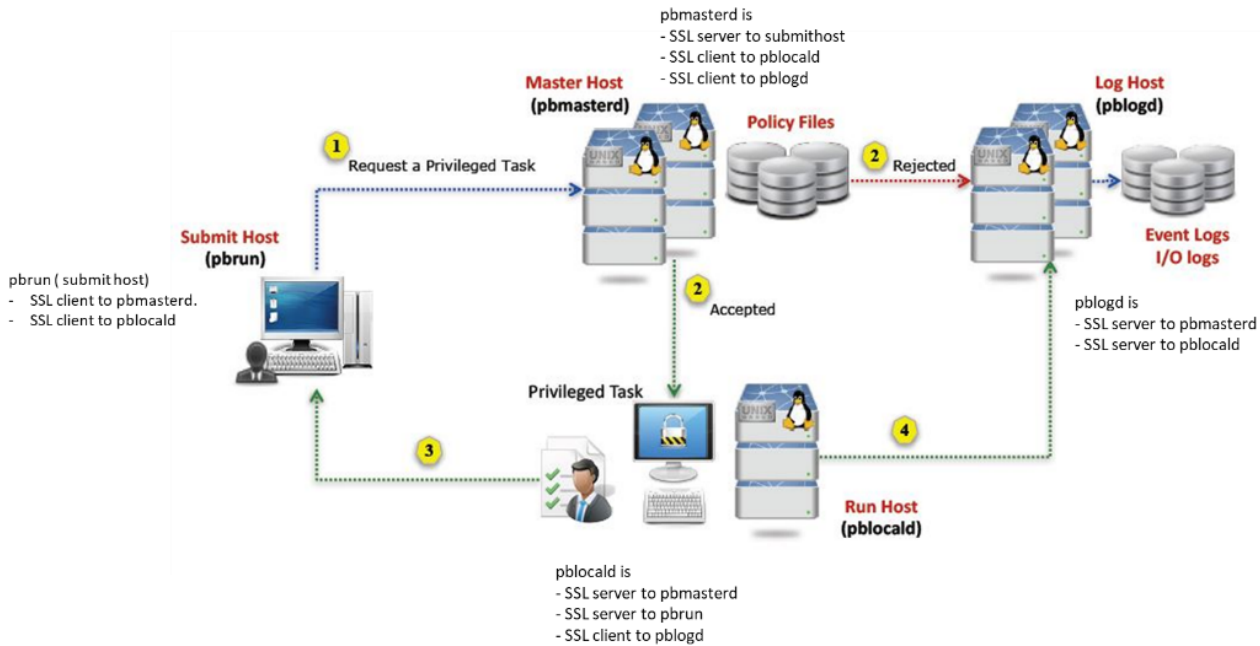
and

```
ssl.openssl.ssl-conf-cmd = (
    "MinProtocol" => " TLSv1",
    "CipherString" => "HIGH:!SSLv2:!3DES:!MD5:!ADH:!AECDH:!DHE:!eNULL:@STRENGTH",
    "Ciphersuites" => "TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA256:TLS_
AES_128_GCM_SHA256"
```

In the following sections the diagram shows the SSL server and SSL client connections between **pbmaterd**, **pblogald**, **pblogd** and **pbrun** and the table shows the required certificate keywords in **pb.settings** file on each host when **validateServer** or **validateClient** is added to **ssloptions**.

SSL Connections in Default Architecture: Classic pbrun

PMUL SSL connections in default architecture: Classic pbrun



On Submithost (pbrun)		On Masterhost (pbmasterd)		On Loghost (pblogd)		On Runhost (pblocald) (submithost != runhost) or (pbrun --di) or (pbmasterd -- disable_optimized_runmode)	
ssloptions	SSL cert/CA file required	ssloptions	SSL cert/CA file required	ssloptions	SSL cert/CA file required	ssloptions	SSL cert/CA file required
validateServer	*sslpbruncafile		sslservercertfile sslserverkeyfile		-		sslservercertfile sslserverkeyfile
-	-	validateServer (pbmasterd is client to pblocald, pblogd)	sslservercafile		sslservercertfile sslserverkeyfile		sslservercertfile sslserverkeyfile

On Submithost (pbrun)		On Masterhost (pbmasterd)		On Loghost (pblogd)		On Runhost (pblocald) (submithost != runhost) or (pbrun --di) or (pbmasterd -- disable_optimized_runmode)	
<i>ssloptions</i>	<i>SSL cert/CA file required</i>	<i>ssloptions</i>	<i>SSL cert/CA file required</i>	<i>ssloptions</i>	<i>SSL cert/CA file required</i>	<i>ssloptions</i>	<i>SSL cert/CA file required</i>
	-		-	validateServer	- (pblogd is not an SSL client to any host)		-
	-		-		sslservercertfile sslserverkeyfile	validateServer	sslservercafile
validateClient	- (pbrun is not an SSL server at any point, also refer to *)		-		-		-
	sslpbruncertfile sslpbrunkeyfile	validateClient	sslservercafile		**sslservercertfile sslserverkeyfile		**sslservercertfile sslserverkeyfile
	-		sslservercertfile sslserverkeyfile	validateClient	sslservercafile		sslservercertfile sslserverkeyfile ***sslpbruncertfile sslpbrunkeyfile
	sslpbruncertfile sslpbrunkeyfile		sslservercertfile sslserverkeyfile		**sslservercertfile sslserverkeyfile	validateClient	sslservercafile

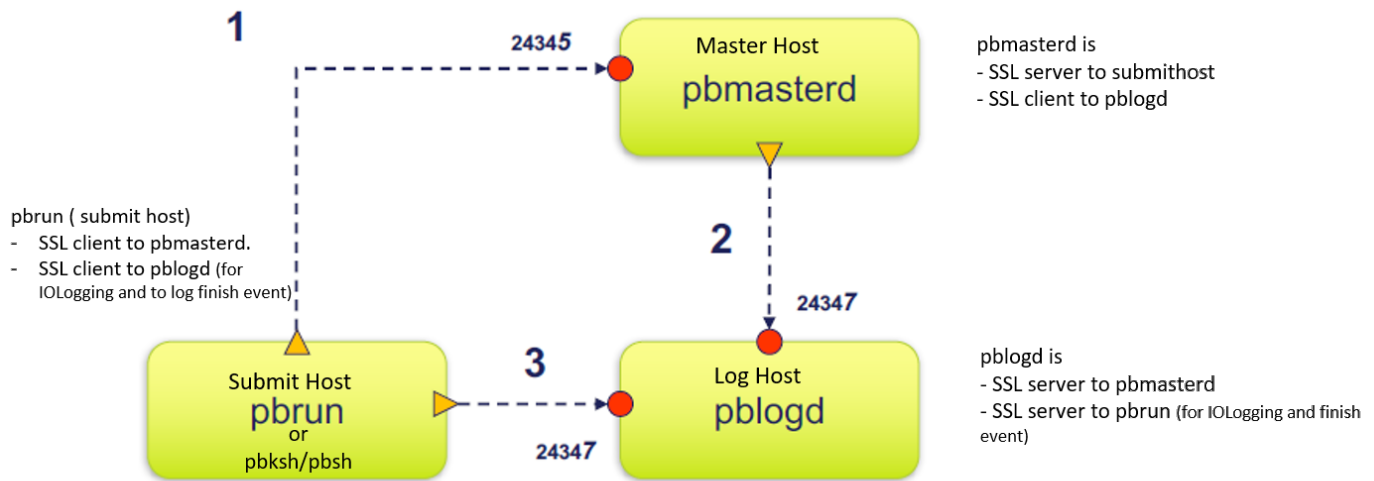
* Mentioning **sslpbruncafile** with or without **validateServer** and **validateClient** options requires **pbmasterd** and **pblocald** certificates.

** Mentioning **sslservercafile** on a SSL client (**pbmasterd** and **pblocald**) with or without **validateServer** and **validateClient** options always requires certificates from its immediate SSL server or servers.

*** **pblocald** needs **sslpbruncertfile** and **sslpbrunkeyfile** to log finish event.

SSL Connections in Optimized Runmode

PMUL SSL connections in optimized runmode



On Submithost (pbrun)		On Masterhost (pbmasterd)		On Loghost (pblogd)	
<i>ssloptions</i>	<i>SSL cert/CA file required</i>	<i>ssloptions</i>	<i>SSL cert/CA file required</i>	<i>ssloptions</i>	<i>SSL cert/CA file required</i>
validateServer	*sslpbruncafile		sslservercertfile sslserverkeyfile		sslservercertfile sslserverkeyfile (for IOLogging and finish event)
-		validateServer (pbmasterd is client to pblogd)	sslservercafile		sslservercertfile sslserverkeyfile
	-		-	validateServer	- (pblogd is not an SSL client to any host)
validateClient	- (pbrun is not an SSL server at any point, also refer to *)		-		-

On Submithost (pbrun)		On Masterhost (pbmasterd)		On Loghost (pblogd)	
<i>ssloptions</i>	<i>SSL cert/CA file required</i>	<i>ssloptions</i>	<i>SSL cert/CA file required</i>	<i>ssloptions</i>	<i>SSL cert/CA file required</i>
	sslpbruncertfile sslpbrunkeyfile	validateClient	sslservercafile		**sslservercertfile sslpbrunkeyfile
	sslpbruncertfile sslpbrunkeyfile (for IOLogging and to log finish event)		sslservercertfile sslserverkeyfile	validateClient	sslservercafile

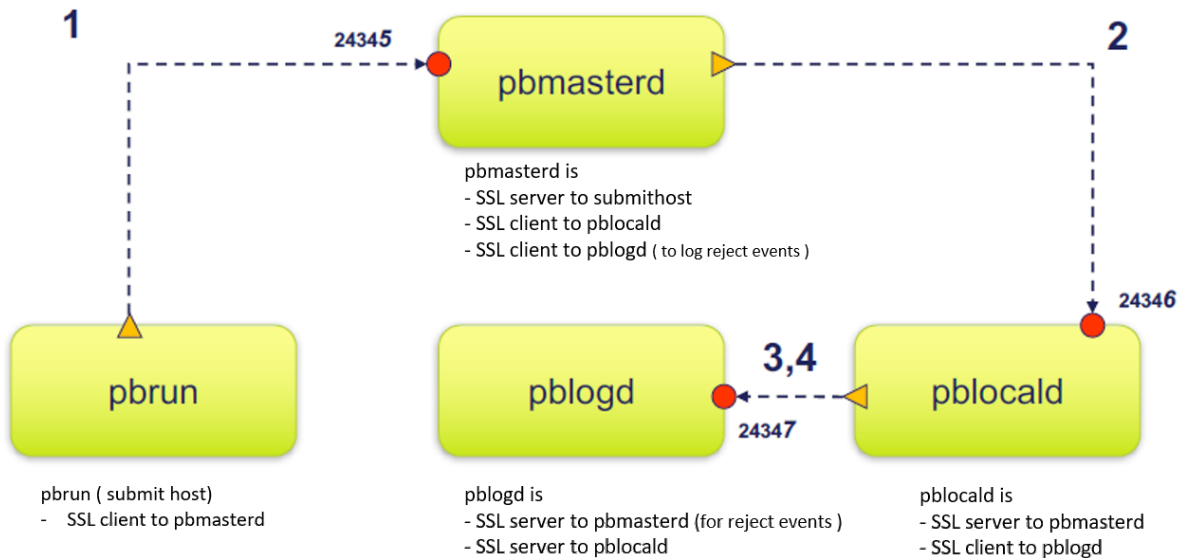
* Mentioning **sslpbruncafile** with or without **validateServer** and **validateClient** options requires **pbmasterd** and **pblogd** certificates.

** Mentioning **sslservercafile** on **pbmasterd** with or without **validateServer** and **validateClient** options always requires certificates from **pblogd**.

SSL Connections with `noreconnect=1` in the Policy

- `noreconnect=1` : **pbrun** does not connect to **pblocald** directly

PMUL SSL connections with `noreconnect=1` in the policy



* `noreconnect=1` : **pbrun** does not connect to **pblocald** directly

On Submithost (pbrun)		On Masterhost (pbmasterd) noreconnect=1 in the policy		On Loghost (pblogd)		On Runhost (pblockd) (submithost != runhost) or (pbrun --di) or (pbmasterd -- disable_optimized_runmode)	
<i>ssloptions</i>	<i>SSL cert/CA file required</i>	<i>ssloptions</i>	<i>SSL cert/CA file required</i>	<i>ssloptions</i>	<i>SSL cert/CA file required</i>	<i>ssloptions</i>	<i>SSL cert/CA file required</i>
validateServer	*sslpbruncafile		sslservercertfile sslserverkeyfile		-		-
-		validateServer (pbmasterd is client to pblockd, pblogd)	sslservercafile		sslservercertfile sslserverkeyfile		sslservercertfile sslserverkeyfile
-			-	validateServer	- (pblogd is not an SSL client to any host)		-
-			-		sslservercertfile sslserverkeyfile	validateServer	sslservercafile
validateClient	- (pbrun is not an SSL server at any point, also refer to *)		-		-		-
	sslpbruncertfile sslpbrunkeyfile	validateClient	sslservercafile		**sslservercertfile sslserverkeyfile		**sslservercertfile sslserverkeyfile
-			sslservercertfile sslserverkeyfile	validateClient	sslservercafile		sslservercertfile sslserverkeyfile ***sslpbruncertfile sslpbrunkeyfile
-			sslservercertfile sslserverkeyfile		**sslservercertfile sslserverkeyfile	validateClient	sslservercafile

* Mentioning **sslpbruncafile** with or without **validateserver** and **validateclient** options requires **pbmasterd** certificates.

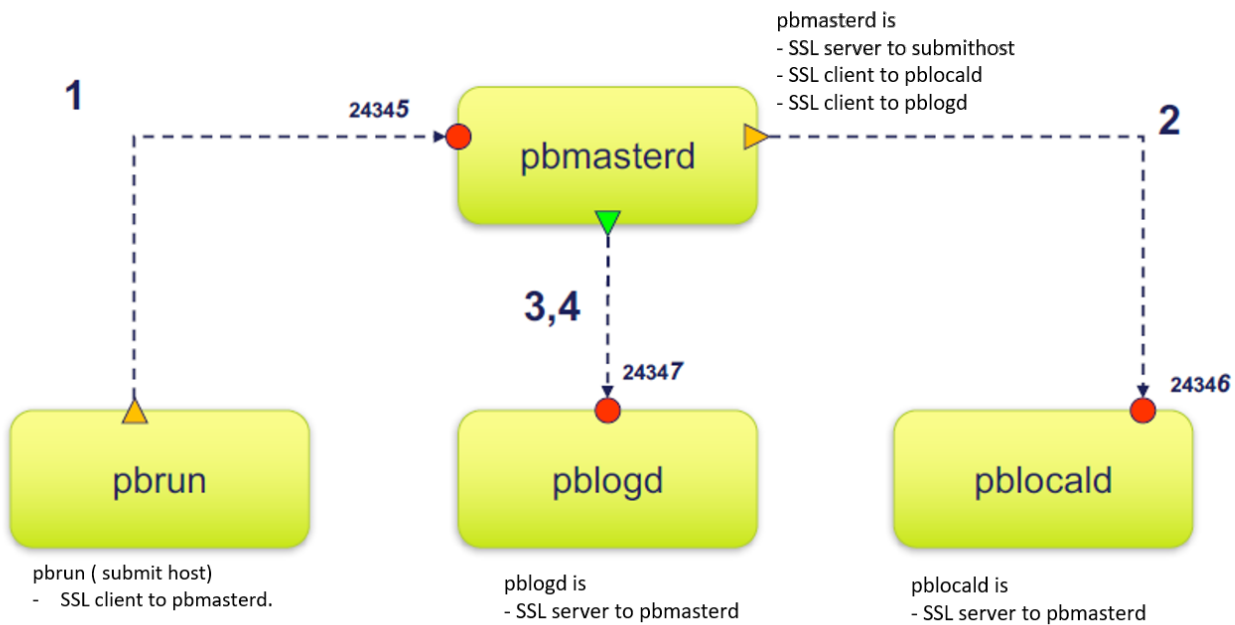
** Mentioning `sslservercafile` on a SSL client (`pbmasterd` and `pblocald`) with or without `validateserver` and `validateclient` options always requires certificates from its immediate SSL server or servers.

*** `pblocald` needs `sslpruncertfile` and `sslprunkeyfile` to log finish event.

SSL Connections with `logignoreconnect=1` in the Policy

- `logignoreconnect=1` : `pblocald` does not connect to `pblogd` directly and `pbrun` does not connect to `pblocald` directly.

PMUL SSL connections with `logignoreconnect=1` in the policy



* `logignoreconnect=1` : `pblocald` does not connect to `pblogd` directly and `pbrun` does not connect to `pblocald` directly

On Submithost (pbrun)		On Masterhost (pbmasterd) logignoreconnect=1 in the policy		On Loghost (pblogd)		On Runhost (pblocald) (submithost != runhost) or (pbrun --di) or (pbmasterd -- disable_optimized_ runmode)	
<i>ssloptions</i>	<i>SSL cert/CA file required</i>	<i>ssloptions</i>	<i>SSL cert/CA file required</i>	<i>ssloptions</i>	<i>SSL cert/CA file required</i>	<i>ssloptions</i>	<i>SSL cert/CA file required</i>
validateServer	*sslpbruncafile		sslservercertfile sslserverkeyfile		-		-
	-	validateServer (pbmasterd is client to pblocald, pblogd)	sslservercafile		sslservercertfile sslserverkeyfile		sslservercertfile sslserverkeyfile
	-		-	validateServer	- (pblogd is not an SSL client to any host)		-
	-		-		-	validateServer	- (pblocald is not a SSL client/server to pblogd as logignoreconnect=1)
validateClient	- (pbrun is not an SSL server at any point, also refer to *)		-		-		-
	sslpbruncertfile sslpbrunkeyfile	validateClient	sslservercafile		**sslservercertfile sslserverkeyfile		**sslservercertfile sslserverkeyfile
	-		sslservercertfilesslserverkeyfile	validateClient	sslservercafile		-
	- (No direct connection between pbrun and pblocald when logignoreconnect=1)		sslservercertfile sslserverkeyfile		-	validateClient	sslservercafile

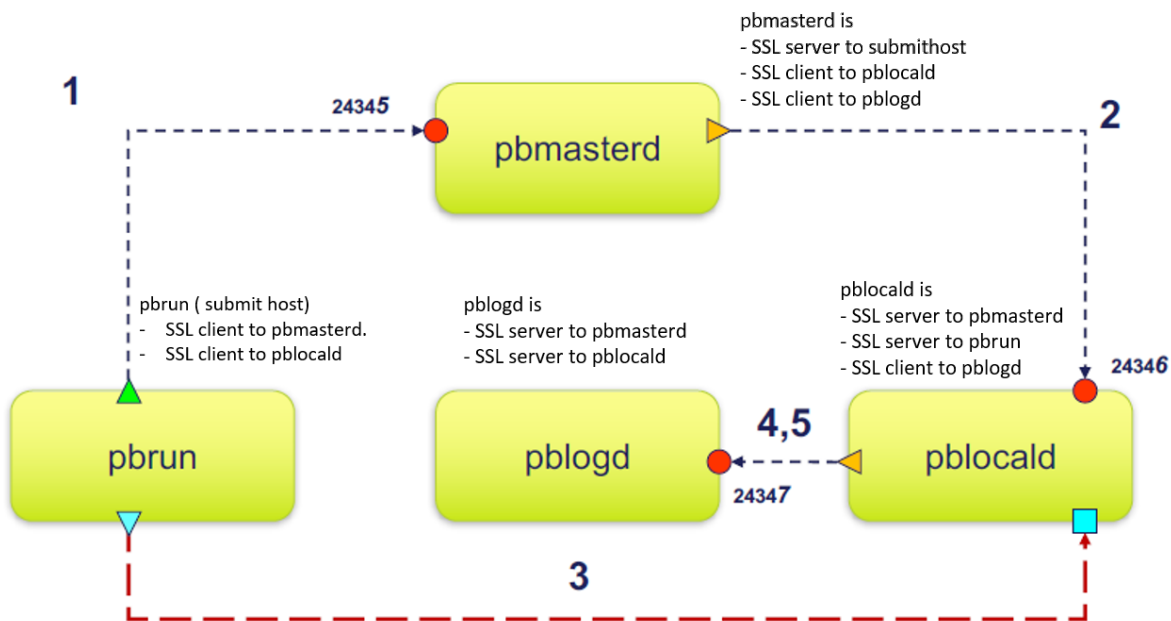
* Mentioning `sslpbruncafile` with or without `validateServer` and `validateClient` options requires `pbmasterd` certificates.

** Mentioning `sslservercafile` on `pbmasterd` with or without `validateserver` and `validateclient` options always requires certificates from `pblocald` and `pblogd`.

SSL Connections with `pbrunreconnection=1` in the Policy

- `pbrunreconnection=1` : `pblocald` listens for the connections that are initiated by `pbrun` under the control of `pbmasterd`.
- `pbrunreconnection=0` : `pbrun` listens for the connections that are initiated by `pblocald` under the control of `pbmasterd`. This value is the default.

PMUL SSL connections with `pbrunreconnection=1` in the policy



* `pbrunreconnection=1` : `pblocald` listens for the connections that are initiated by `pbrun` under the control of `pbmasterd`
`pbrunreconnection=0` : `pbrun` listens for the connections that are initiated by `pblocald` under the control of `pbmasterd`. This value is the default.

On Submithost (pbrun)		On Masterhost (pbmasterd) pbrunreconnection=1 in the policy		On Loghost (pblogd)		On Runhost (pblockd) (submithost != runhost) or (pbrun --di) or (pbmasterd --disable_optimized_runmode)	
<i>ssloptions</i>	<i>SSL cert/CA file required</i>	<i>ssloptions</i>	<i>SSL cert/CA file required</i>	<i>ssloptions</i>	<i>SSL cert/CA file required</i>	<i>ssloptions</i>	<i>SSL cert/CA file required</i>
validateServer	*sslpbruncafile		sslservercertfile sslserverkeyfile		-		sslservercertfile sslserverkeyfile
-		validateServer (pbmasterd is client to pblockd, pblogd)	sslservercafile		sslservercertfile sslserverkeyfile		sslservercertfile sslserverkeyfile
-			-	validateServer	- (pblogd is not an SSL client to any host)		-
-			-		sslservercertfile sslserverkeyfile	validateServer	sslservercafile
validateClient	- (pbrun is not an SSL server at any point, also refer to *)		-		-		-
	sslpbruncertfile sslpbrunkeyfile	validateClient	sslservercafile		**sslservercertfile sslserverkeyfile		**sslservercertfile sslserverkeyfile
-			sslservercertfile sslserverkeyfile	validateClient	sslservercafile		sslservercertfile sslserverkeyfile ***sslpbruncertfile sslpbrunkeyfile
	sslpbruncertfile sslpbrunkeyfile		sslservercertfile sslserverkeyfile		**sslservercertfile sslserverkeyfile	validateClient	sslservercafile

* Mentioning **sslpbruncafile** with or without **validateserver** and **validateclient** options requires **pbmasterd** and **pblockd** certificates.

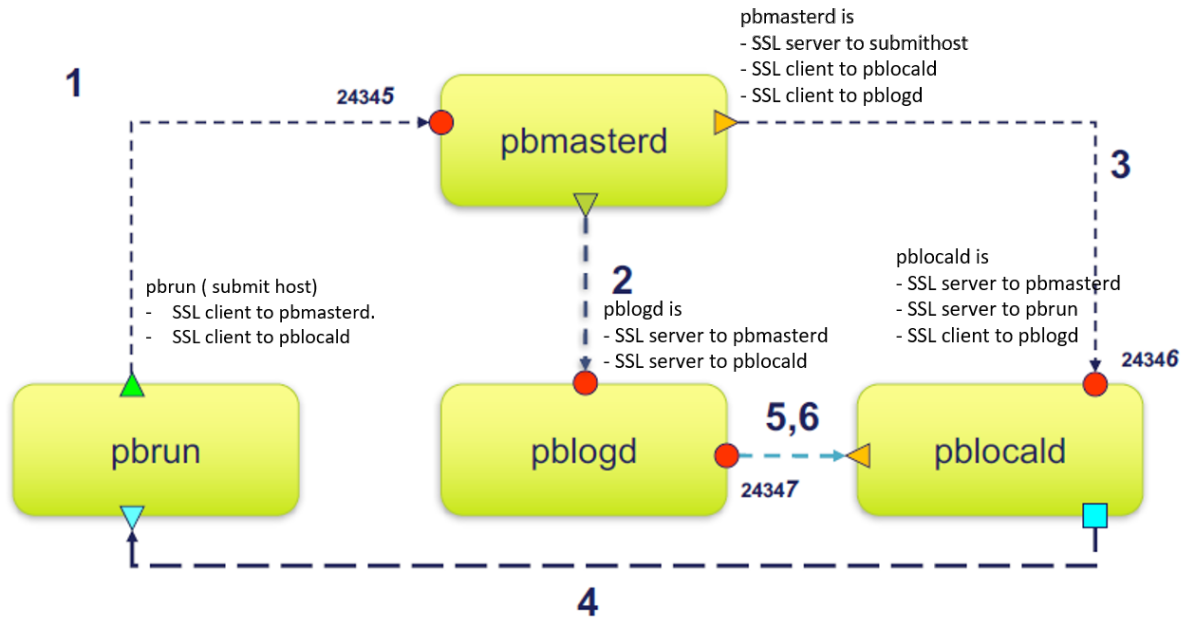
** Mentioning `sslservercafile` on a SSL client (`pbmasterd` and `pblocald`) with or without `validateserver` and `validateclient` options always requires certificates from its immediate SSL server or servers.

*** `pblocald` needs `sslpruncertfile` and `sslprunkeyfile` to log finish event.

SSL Connections with `pblogdreconnection=1` in the Policy

- `pblogdreconnection=1` : `pblocald` listens for the connections that are initiated by `pblogd` under the control of `pbmasterd`.
- `pblogdreconnection=0` : `pblogd` listens for the connections that are initiated by `pblocald` under the control of `pbmasterd`. This value is the default.

PMUL SSL connections with `pblogdreconnection=1` in the policy



* `pblogdreconnection=1` : `pblocald` listens for the connections that are initiated by `pblogd` under the control of `pbmasterd`
`pblogdreconnection=0` : `pblogd` listens for the connections that are initiated by `pblocald` under the control of `pbmasterd`. This value is the default.

On Submithost (pbrun)		On Masterhost (pbmasterd) pblogdreconnection=1 in the policy		On Loghost (pblogd)		On Runhost (pblocald) (submithost != runhost) or (pbrun --di) or (pbmasterd -- disable_optimized_runmode)	
<i>ssloptions</i>	<i>SSL cert/CA file required</i>	<i>ssloptions</i>	<i>SSL cert/CA file required</i>	<i>ssloptions</i>	<i>SSL cert/CA file required</i>	<i>ssloptions</i>	<i>SSL cert/CA file required</i>
validateServer	*sslpbruncafile		sslservercertfile sslserverkeyfile		-		sslservercertfile sslserverkeyfile
-		validateServer (pbmasterd is client to pblocald, pblogd)	sslservercafile		sslservercertfile sslserverkeyfile		sslservercertfile sslserverkeyfile
-			-	validateServer	- (pblogd is not an SSL client to any host)		-
-			-		sslservercertfile sslserverkeyfile	validateServer	sslservercafile
validateClient	- (pbrun is not an SSL server at any point, also refer to *)		-		-		-
	sslpbruncertfile sslpbrunkeyfile	validateClient	sslservercafile		**sslservercertfile sslserverkeyfile		**sslservercertfile sslserverkeyfile
-			sslservercertfile sslserverkeyfile	validateClient	sslservercafile		sslservercertfile sslserverkeyfile ***sslpbruncertfile sslpbrunkeyfile
	sslpbruncertfile sslpbrunkeyfile		sslservercertfile sslserverkeyfile		**sslservercertfile sslserverkeyfile	validateClient	sslservercafile

* Mentioning **sslpbruncafile** with or without **validateServer** and **validateClient** options requires **pbmasterd** and **pblocald** certificates.

** Mentioning **sslservercafile** on a SSL client (**pbmasterd** and **pblocald**) with or without **validateserver** and **validateclient** options always requires certificates from its immediate SSL server or servers.

*** **pblocald** needs **sslpruncertfile** and **sslprunkeyfile** to log finish event.

Kerberos Version 5

Privilege Management for Unix and Linux can use Kerberos v5 to authenticate its various parts and to exchange encryption key information.

To use Kerberos with Privilege Management for Unix and Linux, you must register **pbmasterd**, **pblocald**, and **pblogd** as Kerberos principals. The principals should look like this (substitute your own host and **pblogd** principal names):

- **pbmasterd/kerberizedmachine.your_realm.com**
- **pblocald/kerberizedmachine.your_realm.com**
- **pblogd/kerberizedmachine.your_realm.com**

These principals must be added to the **keytab** file. Users also need to be principals and need a target.

The default principals are **pbmasterd**, **pblocald**, and **pblogd**. These can be overridden by the **mprincipal**, **lprincipal**, and **gprincipal** settings in the settings file.

All Privilege Management for Unix and Linux client and server programs can use Kerberos Version 5 for authentication and session encryption keys.

Privilege Management for Unix and Linux clients request verification to use **pbmasterd** by checking the submitting user ticket cache or obtaining a ticket for **pbmasterd** with the principal in **mprincipal**.

Privilege Management for Unix and Linux daemons request verification to access other daemons by checking the services' principals for both daemons, as listed in the following table.

Kerberos Principal Usage

From	Principal	Connection Type	To	Principal
pbrun pbksh pbsh pbguid	user@realm	Direct	pbmasterd	mprincipal/ masterhost@real m
pbrun pbksh pbsh	user@realm	Dynamic	pblocald	lprincipal/ runhost@realm
pbrun pbksh pbsh pblocald	principal/ runhost@realm	Direct or dynamic	pblogd	gprincipal/ loghost@realm
pbmasterd	mprincipal/ masterhost@real m	Direct	pblocald	lprincipal/ loghost@realm
pbmasterd	mprincipal/ masterhost@real m	Direct	pblogd	gprincipal/ loghost@realm

kerberos

- **Version 4.0.0 and later:** **kerberos** setting available.
- **Version 22.1 and later:** **krbfirstpbmasterd** is available. The setting **kerberos** is a list setting. Use **getlistsetting()** in policy language to get the settings list.

When set to **yes**, the **kerberos** setting enables the use of the Privilege Management for Unix and Linux Kerberos Version 5 features. When set to **no**, the **kerberos** setting disables the use of these features.

Use the option **krbfirstpbmasterd** with the option **yes** to enable the compatibility of 10.3.2 or below versions of clients with newer versions of PMUL servers and clients.



Note: The **krbfirstpbmasterd** option is valid from version 22.1. The **krbfirstpbmasterd** option is enabled on all newer versions of endpoints (22.1+ servers and clients. No changes are required on 10.3.2 or below versioned clients).



Example:

```
kerberos yes
kerberos yes krbfirstpbmasterd
```

Default

```
kerberos no
```

Used on

- Log hosts
- Policy server hosts
- Submit hosts
- Run hosts

keytab

Version 4.0.0 and later: **keytab** setting available.

The **keytab** setting contains the name of the Kerberos 5 Key Table. Newer versions of Kerberos discourage setting the keytab in this fashion and favor using the **krb5.conf** file or the **KRB5_KTNAME** environment variable. Use of the **keytab** setting should be avoided.

keytabencryption

- **Version 8.0.0 and earlier:** **keytabencryption** setting not available.
- **Version 8.0.1 and later:** **keytabencryption** setting available.

The **keytabencryption** setting specifies which cipher all Privilege Management for Unix and Linux components use for Kerberos negotiations. The algorithm must match the default algorithm used by the Kerberos server. Supported values include **des-hmac**, **des3-hmac**, and **arcfour-hmac**. As of this writing, the AES algorithms are not supported, which effectively limits using Active Directory as a Kerberos server. This keyword is mandatory to support more recent, non-DES Kerberos implementations because Privilege Management for Unix and Linux cannot automatically determine the best cipher.

**Example:**

```
keytabencryption arcfour-hmac
```

Default

```
keytabencryption des-hmac
```

Used on

- Log hosts
- Policy server hosts
- Submit hosts
- Run hosts

gprincipal

- **Version 4.0.0 and later:** **gprincipal** setting available.

The **gprincipal** setting contains the principal that the policy server daemon (**pbmasterd**), the local daemon (**pblocald**) and clients that are running in local mode (for example, **pbrun -l ...**) use to verify access to the log server daemon (**pblogd**). The host name and realm are appended to form the full principal.

**Example:**

```
gprincipal pblogd_principal
```

Default

```
gprincipal pblogd
```

Used on

- Log hosts
- Policy server hosts

- Submit hosts
- Run hosts

lprincipal

- **Version 4.0.0 and later:** lprincipal setting available.

The **lprincipal** setting contains the principal that the policy server daemon (**pbmasterd**) and client programs use to verify access to the local daemon (**pblocald**). The host name and realm are appended to form the full principal.



Example:

```
lprincipal pblocald_principal
```

Default

```
lprincipal pblocald
```

Used on

- Log hosts
- Policy server hosts
- Run hosts
- Submit hosts

mprincipal

- **Version 4.0.0 and later:** mprincipal setting available.

The **mprincipal** setting contains the principal that the Privilege Management for Unix and Linux clients use to verify access to the policy server daemon (**pbmasterd**). The host name and realm are appended to form the full principal.



Example:

```
mprincipal pbmasterd_principal
```

Default

```
mprincipal pbmasterd
```

Used on

- Policy server hosts
- Submit hosts

sprincipal

- **Version 5.2 and earlier:** **sprincipal** setting not available.
- **Version 6.0 and later:** **sprincipal** setting available.

The **sprincipal** setting contains the principal that the Privilege Management for Unix and Linux **pbsync** client uses to verify access to the log synchronization daemon (**pbsyncd**). The host name and realm are appended to form the full principal.



Example:

```
sprincipal pbsync_principal
```

Default

```
sprincipal pbsyncd
```

Used on

- Log hosts
- Policy server hosts
- Sync hosts

kerberosvalidatecacheuser

- **Version 4.0 and later:** **kerberosvalidatecacheuser** setting available.

If set to **yes**, compares current user's Unix/Linux username with the Kerberos client name. If they do not match, it invalidates the cache and new credentials have to be provided.

Default

```
kerberosvalidatecacheuser no
```

Privilege Management for Unix and Linux Shared Libraries

When configured with Kerberos, SSL, LDAP, or PAM, Privilege Management for Unix and Linux requires the appropriate third-party libraries. The installation provides Kerberos, SSL, LDAP, and PAM libraries that are designed to work with Privilege Management for Unix and Linux. We recommend you install these third-party libraries.



Note: The shared libraries for the following operating systems are not currently supported:

- NCR
- IRIX
- OSF
- QNX

Shared Libraries for Kerberos

These settings are related to the shared libraries that are needed for Kerberos in Privilege Management for Unix and Linux.

sharedlibkrb5dependencies

- **Version 5.0.4 and earlier:** `sharedlibkrb5dependencies` setting not available.
- **Version 5.1.0 and later:** `sharedlibkrb5dependencies` setting available.

The libraries are listed in the order they are loaded (dependencies first). This setting should be used in either of the following circumstances:

- The `kerberos` setting is set to **yes**, the `pam` setting is set to **yes**, and PAM uses Kerberos
- The `ssl` setting is set to **yes** and the SSL libraries that are listed in the `sharedlibssldependencies` setting are dependent on the Kerberos libraries

By default, the shared libraries that are listed for this setting are the ones that are shipped with Privilege Management for Unix and Linux. However, you can replace them with libraries that are used by the PAM or SSL services that are installed on the Privilege Management for Unix and Linux host computer.



Example:

```
sharedlibkrb5dependencies /usr/lib/beyondtrust/pb/libcom_err.so.3
/usr/lib/beyondtrust/pb/libk5crypto.so.3.1 /usr/lib/beyondtrust/pb/libkrb5.so.3.3
/usr/lib/beyondtrust/pb/libgssapi_krb5.so.2.2
```

Default

No default value

Used on

- Log hosts
- Policy server hosts
- Submit hosts
- Run hosts

loadkrb5libs

- **Version 5.2 and earlier:** `loadkrb5libs` setting not available.
- **Version 6.0 and later:** `loadkrb5libs` setting available.

The `loadkrb5libs` setting determines whether the libraries that are listed in the `sharedlibkrb5dependencies` setting are loaded at runtime even if the value of the `kerberos` setting is `no`. This setting is ignored when `kerberos` is set to `yes`. This setting is useful in certain cases where the operating system is configured to use Kerberos and the Privilege Management for Unix and Linux `submitconfuser()` function returns false even when the correct Kerberos password is supplied.



Example:

```
loadkrb5libs yes
```

Default

```
loadkrb5libs no
```

Used on

- Log hosts
- Policy server hosts
- Submit hosts
- Run hosts

Shared Libraries for SSL

The following setting is related to the shared libraries needed for SSL use.

loadssllibs

- **Version 6.2.5 and earlier:** `loadssllibs` setting not available.
- **Version 6.2.6 and later:** `loadssllibs` setting available.

The **loadssllibs** setting determines whether the libraries that are listed in the **sharedlibssldependencies** setting are loaded at runtime even if the value of the **ssl** setting is **no**. This setting is ignored when **ssl** is set to **yes**. This setting is useful in certain cases where the operating system is configured to use SSL and we need to force Privilege Management for Unix and Linux to load the SSL libraries.

**Example:**

```
loadssllibs yes
```

Default

```
loadssllibs no
```

Used on

- Log hosts
- Policy server hosts
- Submit hosts
- Run hosts

sharedlibssldependencies

- **Version 5.0.4 and earlier:** **sharedlibssldependencies** setting not available.
- **Version 5.1.0 and later:** **sharedlibssldependencies** setting available .

The libraries are listed in the order they are loaded (dependencies first). This setting should be used in either of the following circumstances:

- The **ssl** setting is set to **yes**
- LDAP is used in the policy or by PAM and the LDAP libraries that are listed in the **sharedliblddependencies** setting are dependent on the SSL libraries

By default, the shared libraries listed for this setting are shipped with Privilege Management for Unix and Linux. However, you can replace them with libraries that are used by the SSL service that is installed on the Privilege Management for Unix and Linux host computer.

**Example:**

```
sharedlibssldependencies /usr/lib/beyondtrust/pb/libcrypto.so.1.1  
/usr/lib/beyondtrust/pb/libssl.so.1.1
```

Default

No default value

Used on

- Log hosts
- Policy server hosts
- Submit hosts
- Run hosts

Shared Libraries for LDAP

The following setting is related to the shared libraries that are needed for LDAP use.

loadldaplibs

- **Version 6.2.5 and earlier:** `loadldaplibs` setting not available.
- **Version 6.2.6 and later:** `loadldaplibs` setting available.

The `loadldaplibs` setting determines whether the libraries that are listed in the `sharedlibldapdependencies` setting are loaded at runtime even if policy LDAP functions are not used. This setting is useful in certain cases where the operating system is configured to use LDAP and we need to force Privilege Management for Unix and Linux to load the LDAP libraries.



Example:

```
loadldaplibs yes
```

Default

```
loadldaplibs no
```

Used on

- Log hosts
- Policy server hosts
- Submit hosts
- Run hosts

sharedlibldapdependencies

- **Version 5.0.4 and earlier:** `sharedlibldapdependencies` setting not available.
- **Version 5.1.0 and later:** `sharedlibldapdependencies` setting available .

The libraries are listed in the order they are loaded (dependencies first). This setting should be used in either of the following circumstances:

- LDAP is used in the Privilege Management for Unix and Linux policy
- The **pam** setting is set to **yes** and PAM is using LDAP

By default, the shared libraries that are listed for this setting are shipped with Privilege Management for Unix and Linux. However, you can replace them with libraries that are used by the LDAP service that is installed on the Privilege Management for Unix and Linux host computer.



Example:

```
sharedlibldapdependencies /usr/lib/beyondtrust/pb/liblber-2.5.so.0.1.7
/usr/lib/beyondtrust/pb/libldap-2.5.so.0.1.7
```

Default

No default value

Used on

- Log hosts
- Policy server hosts
- Submit hosts
- Run hosts

Shared Library Directory Location for AIX and HP (PA RISC)

For AIX and HP (PA-RISC), the directory for installing third-party libraries must be in one of the following locations:

- **/usr/lib/symark/pb**
- **/usr/lib**
- **/lib**
- **/usr/local/lib**

If any other directory is specified, then it is rejected with an error message stating that you must use one of these four directory locations.

Shared Library File Name for AIX

The notation that is used on AIX to specify some libraries (Kerberos and LDAP) is different from other platforms. On AIX for third-party libraries that are archives, you also need to specify the shared object that is a member of the archive and add it to the file name.



Example: If *libcom_err.a.3.0* is an archive and *shr.0.3.0* is the actual shared object, then the file specification for the member of the archive is:

```
libcom_err.a.3.0(shr.0.3.0)
```



Note: For SSL, because the library is not an archive, it is not necessary to alter the file name.

Shared Libraries for PAM

The following are the shared libraries needed for PAM use.

libpam

- **Version 5.2 and earlier:** libpam setting not available.
- **Version 6.0 and later:** libpam setting available.

libpam is a user-defined PAM library that Privilege Management for Unix and Linux uses as a first option in case the system does not use the standard default PAM libraries. The notation used for AIX to specify the OS-provided PAM library is the following:

```
/usr/lib/libpam.a(shr.o)
```



Example:

```
libpam /lib/libpam.so.1
```

Default

No default value

Used on

- Policy server hosts
- Submit hosts
- Run hosts
- Policy server hosts
- Submit hosts
- Run hosts

Hardware Security Module (HSM)

Privilege Management for Unix and Linux, through its integration with the SafeNet Luna SA Hardware Security Module (HSM), provides the first privileged user management solution to use FIPS 140-2 Security Level 2-validated key storage services to achieve compliance with the most strict key storage requirements and standards. Privilege Management for Unix and Linux supports the configuration of an SSL engine. An SSL engine is a plug-in mechanism for third parties to add extra cryptographic capabilities to SSL. The SSL engine must be properly configured according to the engine provider's instructions. The SSL library that is shipped with Privilege Management for Unix and Linux does not support the use of SSL engines. Therefore, to use an SSL engine, you must build your own set of SSL libraries to

support the SSL engine. If you use Kerberos or LDAP, then you must also build your own set of those libraries. The file name of the SSL engine shared object should be appended to the **sharedlibssldependencies** setting, and the engine ID should be specified using the **sslengine** keyword.

sslengine

- **Version 5.0.4 and earlier:** **sslengine** setting not available.
- **Version 5.1.0 and later:** **sslengine** setting available.

The **sslengine** setting specifies the SSL engine ID to be used with the HSM. The value is case-sensitive.



Example: The following is an example **pb.settings** configuration when using the SafeNet Luna SA Hardware Security Module:

```
sharedlibkrb5dependencies none
sharedlibldapdependencies none
sharedlibssldependencies /usr/local/lunassl/lib/libcrypto.so.0.9.8
/usr/local/lunassl/lib/libssl.so.0.9.8
/usr/local/lunassl/lib/engines/liblunaca3.so

ssl yes
sslservercertfile /etc/pb/CERTS/safenet.crt
sslserverkeyfile /etc/pb/CERTS/safenet.key

sslengine LunaCA3
```

*New SSL libraries with engine support are built and installed in the **/usr/local/lunassl** directory. Kerberos and LDAP are not in use. The engine ID is **LunaCA3**. The key file value is a name that is interpreted by the engine to access the private key on the HSM.*

Default

No default value

Used on

- Policy server hosts
- Log hosts

GUI Configuration

These settings control where the Privilege Management for Unix and Linux GUI looks for its help files and user default settings.

builderdir

- **Version 4.0.0 and later:** **builderdir** setting available.

The **builderdir** setting specifies where **pbguid** looks for its help files.



Example:

```
builderdir /usr/local/pbguid/help_files
```

Default

```
builderdir /usr/local/lib/<prefix>pbbuilder<suffix>
```

Used on

GUI hosts

guidefaults

- **Version 4.0.0 and later:** **guidefaults** setting available.

Pbguid maintains a file of user defaults, which include the starting directory for logs, screen colors, etc. The location of this file is specified in the **guidefaults** setting.



Example:

```
/pbguid.defaults
```

Default

No default value

Used on

GUI hosts

guireplaytimelimit

- **Version 9.4.1 and earlier:** `guireplaytimelimit` setting not available.
- **Version 9.4.3 and later:** `guireplaytimelimit` setting available.

If an I/O log was archived from the original location on the log host using the log archiving feature of Privilege Management for Unix and Linux, PMUL GUI on that logserver can still replay the I/O log. REST API is used by **pbguid** to retrieve a temporary copy from the archive server which is then read for replay. For security purposes, the copy of the log is only available for the time defined (in minutes) by `guireplaytimelimit`.



Example:

```
guireplaytimelimit 20
```

Default

```
guireplaytimelimit 30
```

Used On

GUI hosts

Other Security Issues

runsecurecommand

- **Version 4.0.0 and later:** `runsecurecommand` setting available.

The `runsecurecommand` setting enables the administrator to perform an extra check on the security of the requested command. This check helps to ensure that someone other than `root` or the `runuser` (for example, `sys` or `oracle`) could not have compromised the command.

When set to `yes`, the `runcommand` and all directories above it are checked to determine if anyone other than `root` or the `runuser` has write permission. If the command file or any of the directories above it are writable by anyone other than `root` or the `runuser`, then the run host refuses to run the command. The policy language variable `runsecurecommand` can be set to `true` by the configuration policy on the policy server host for the same effect.



Note: This keyword does not apply to `pbssh`. If it is present in the settings file, it does not have any effect on `pbssh` and is ignored.



Example:

```
runsecurecommand yes
```

Default

```
runsecurecommand no
```

Used on

- Run hosts
- Submit hosts, when using local mode

rejectnullpasswords

- **Version 4.0.0 and later:** `rejectnullpasswords` setting available.

Some systems allow the use of null passwords in their password databases. When null passwords are allowed, a carriage-return at a password prompt matches that null password. If you want to always reject attempts to enter a password for an account with a null password, you can set `rejectnullpasswords` to `yes`.


Example:

```
rejectnullpasswords yes
```

Default

```
rejectnullpasswords no
```

Used on

- Policy server hosts
- Submit hosts
- Run hosts

enforceRunCwd

- **Version 5.0.2 and earlier:** `enforceRunCwd` setting not available.
- **Version 5.0.3 and later:** `enforceRunCwd` setting available.

The `enforceRunCwd` setting enforces the `runcwd` when set to **yes** or when it is not set. When set to **yes** and the user does not have permission for the `runcwd`, the task is rejected. When the secured task cannot change to the `runcwd` directory (because of bad permissions, or because the directory does not exist), then the `enforceRunCwd` setting determines whether the secured task should be run from `/tmp`, or whether it should be denied.



Note: This keyword does not apply to `pbssh`. If it is present in the settings file, it does not have any effect on `pbssh` and is ignored.

Syntax

```
enforceRunCwd <yes|no>
```

Valid Values

- **yes:** Enforce the `runcwd` and do not run the command in `/tmp`.
- **no:** Revert to the old behavior and run the command in `/tmp`.


Example:

```
enforceRunCwd yes
```

Default

```
enforceRunCwd yes
```

Used on

Run hosts


warnuseronerror

- **Version 4.0.0 and later:** `warnuseronerror` setting available.

Privilege Management programs, such as `pbrun`, `pblogd`, and `pbmasterd`, can produce diagnostic message about security problems. These messages include file systems that are writable, the `pb.conf` file being writable, and so forth.

Because a user might be able to use that information to damage a system, the full diagnostic messages are recorded only in the log files. The user sees the generic message, *Security error, please see your administrator*.

To enable the user see the full diagnostic messages, set `warnuseronerror` to `yes`.

 **Note:** There is a limitation to this setting. When an error about the security of the settings file occurs, the user is never notified.

Example:

```
warnuseronerror yes
```

Default

```
warnuseronerror no
```

Used on

- Policy server hosts
- Run hosts
- Submit hosts

showunsecurewarnings

- **Version 5.1.1 and earlier:** `showunsecurewarnings` setting not available.
- **Version 5.1.2 and later:** `showunsecurewarnings` setting available.

Privilege Management programs, such as **pbrun** and **pbmasterd**, can produce diagnostic message about security problems. These messages include information about licensing files and expiration.

This setting supersedes the value of **warnuseronerror** only if the messages do not pose a security risk. When **showunsecurewarnings** is enabled, all messages that can safely be displayed on the client system are displayed. Display of secure messages still depends on the value of **warnuseronerror**.

To allow the user to see the **unsecure** diagnostic messages, set **showunsecurewarnings** to **yes**.

**Example:**

```
showunsecurewarnings yes
```

Default

No default value

Used on

- Policy server hosts
- Run hosts
- Submit hosts

clientdisableoptimizedrunmode

- **Version 5.2 and earlier:** **clientdisableoptimizedrunmode** setting not available.
- **Version 6.0 and later:** **clientdisableoptimizedrunmode** setting available.

Privilege Management for Unix and Linux optimized run mode feature enables a task to be run on the submit host after being validated by the Policy Server host, without invoking **pblocald**.

When set to **yes**, the **clientdisableoptimizedrunmode** setting disables optimized run mode for all **pbrun** invocations on the affected host. This setting is equivalent to invoking **pbrun** with the **--disable_optimized_runmode** command line option.

**Example:**

```
clientdisableoptimizedrunmode yes
```

Default

```
clientdisableoptimizedrunmode no
```

Used on

Submit hosts

i For more information, please see the following:

- ["Optimized Run Mode Processing" on page 21](#)
- ["pbrun" on page 465](#)
- ["masterdisableoptimizedrunmode" on page 261](#)

masterdisableoptimizedrunmode

- **Version 5.2 and earlier:** `masterdisableoptimizedrunmode` setting not available.
- **Version 6.0 and later:** `masterdisableoptimizedrunmode` setting available.

Privilege Management for Unix and Linux optimized run mode feature enables a task to be run on the submit host after being validated by the policy server host, without invoking `pblocald`.

When set to **yes**, the `clientdisableoptimizedrunmode` setting disables optimized run mode for all `pbrun` invocations that are accepted by the affected policy server host. This setting is equivalent to invoking `pbmasterd` with the `--disable_optimized_runmode` command line option.



Example:

```
masterdisableoptimizedrunmode yes
```

Default

```
masterdisableoptimizedrunmode no
```

Used On

Policy server hosts

i For more information, please see the following:

- ["Optimized Run Mode Processing" on page 21](#)
- ["pbmasterd" on page 441](#)
- ["clientdisableoptimizedrunmode" on page 260](#)

execute_via_su

- **Version 7.0 and earlier:** `execute_via_su` setting not available.
- **Version 7.1.0 and later:** `execute_via_su` setting available.

The run environment for the secured task is normally dictated by the policy server policy. It may be desirable to have the runhost dictate the run environment for the secured task. Privilege Management for Unix and Linux v7.1 and above can use the **su** - command to create a login shell for the secured task, thus allowing the login mechanism to setup the run environment. The policy server host keyword **execute_via_su** in **/etc/pb.settings** globally enables using **su** - to execute the secured task.

This keyword can be overridden by the policy variable with the same name **execute_via_su**. The **execute_via_su** variable's initial value is based on the keyword setting's value. When **execute_via_su** is used, any run environment setup in the policy affects the execution of **su** - rather than the execution of the secured task. This includes the use of **runcwd**, **setenv()**, **keepenv()**, etc as well as **!g!**, **!G!**, etc. Entitlement reports do not indicate that **su** - is used, however the Accept events in the event log show that **su** - is used to invoke the secured task.



Note: This feature does not work for **runusers** whose login is disabled (for example, via **/sbin/nologin** or **/bin/false**).



Note: On some operating systems, the **su** program does not pass the **tty** through to the command executed. The **execute_via_su** feature should not be used with secured tasks that require a **tty** on those operating systems.

Keyword/Policy Variable Hierarchy

Settings Keyword	Policy Variable	Result uses su -?
unset	unset	no
unset	TRUE	YES
unset	FALSE	no
No	unset	no
No	TRUE	YES
No	FALSE	no
Yes	unset	YES
Yes	TRUE	YES
Yes	FALSE	no



Example:

```
execute_via_su yes
```

Default

```
execute_via_su no
```

Used On

Policy server hosts

credentialtimeout

Use the command **pbadmin --auth --login** to cache credentials to facilitate working with remote services. The **credentialtimeout** setting is the maximum length of time (in seconds) that the authentication credential is cached.



Example:

```
credentialtimeout 900
```

Default

```
credentialtimeout 1800
```



For more information, please see "[Authentication Credential Cache Options](#)" on page 386 .

logfilepermissions

The **logfilepermissions** setting specifies the permissions that Privilege Management for Unix and Linux uses when creating certain files such as generated pbreport files, I/O logs, and flat file event logs. The default permission is **600** and you can not specify permission less secure than **644**.

Default

```
logfilepermissions 600
```

Used On

All servers

Licensing

Privilege Management for Unix and Linux synchronizes license and statistical data among all its servers. The settings described in this section allow the administrator to make some adjustments to the synchronization process.

 For more information about licensing, please see "[License Management](#)" on page 286.

pblicenserefresh

- **Version 9.3.0 and earlier:** `pblicenserefresh` setting not available.
- **Version 9.4.0 and later:** `pblicenserefresh` setting available.

The primary license server maintains the master copy of the license database which holds information about the license and service utilization by clients.

For efficiency, the secondary license servers and other servers that provide service maintain a local copy of the license database. It is periodically updated with data retrieved from the primary license server.

The `pblicenserefresh` option defines the interval in seconds between each request to the primary license server to refresh the local license data.

Example:

```
pblicenserefresh 86400
```

Default

```
pblicenserefresh 300
```

Used On

All servers

pblicenseretireafter

- **Version 9.3.0 and earlier:** `pblicenseretireafter` setting not available.
- **Version 9.4.0 and later:** `pblicenseretireafter` setting available.

By default, the **AutoRetire** attribute of the license string defines the number of consecutive days that a client can be inactive before it is auto retired.

The `pblicenseretireafter` option allows the administrator to prolong the auto retire interval by setting it to a number larger than the **AutoRetire** license attribute.

The default value of `pblicenseretireafter` is 0, which ensures that the license string's **AutoRetire** default interval is used.



Note: An auto retired client still uses a license slot unless it is put in long-term retirement. This can be done either manually using `pbadm` program, or automatically using the `freelicenseofautoretiredhosts` setting.



Example:

```
pblicenseretireafter 90
```

Default

```
pblicenseretireafter 0
```

Used On

Primary License Server

licensehistory

- **Version 9.4.6 and earlier:** `licensehistory` setting not available.
- **Version 10.0.0 and later:** `licensehistory` setting available.

The `licensehistory` setting is configured on the primary license server, and is synchronized to all hosts that provide a service. If it is enabled, every license event is logged to the primary log server for detailed license information.

When `licensehistory` is set to **yes**, a new record is added to `pbevent.db` for every `pbrun` (or any other "client"), keeping track of every access on each host. This can grow the size of `pbevent.db`. Only enable `licensehistory` if you want every single access to the license database logged.



Example:

```
licensehistory yes
```

Default

```
licensehistory no
```

Used On

All servers

licenseservers

- **Version 9.4.6 and earlier:** `licenseservers` setting not available.
- **Version 10.0.0 and later:** `licenseservers` setting available.

The `licenseservers` setting details those hosts that are license servers. The primary license server is first in the list, with subsequent secondary license servers listed, in order of failover, afterwards. If Registry Name Service is configured this value should be an asterisk, (*) denoting that the value is held within the service database. This setting should be consistent across Privilege Management for Unix and Linux, and is synchronized from the primary license server to other servers.



Example:

```
licenseservers myhost1 myhost2
```

Default

No default value

Used On

All servers

licensestatsdb

- **Version 9.4.6 and earlier:** `licensestatsdb` setting not available.
- **Version 10.0.0 and later:** `licensestatsdb` setting available.

The `licensestatsdb` allows the specification of an absolute or relative path to the license database on server installations. If the path is relative, the absolute path is calculated using the `databasedir` setting. All of the license information, including the license itself, and client and service statistics, are stored in the database.



Example:

```
licensestatsdb /mypath/pblicense.db
```

Default

```
licensestatsdb /opt/<prefix>pbul<suffix>/dbs/pblicense.db
```

Used On

All servers

licensestatswq

- **Version 9.4.6 and earlier:** `licensestatswq` setting not available.
- **Version 10.0.0 and later:** `licensestatswq` setting available.

While processing license statistics or logging license events temporary files are created to increase performance. These files are created with names derived from the `licensestatswq` setting.



Example:

```
licensestatswq /mypath/pblicense.wq
```

Default

```
licensestatswq /opt/<prefix>pbul<suffix>/dbs/pblicense.wq
```

Used on

All servers

licensestatswqnum

- **Version 9.4.6 and earlier:** `licensestatswqnum` setting not available.
- **Version 10.0.0 and later:** `licensestatswqnum` setting available.

While processing license statistics or logging license events, temporary files are created to increase performance. This setting specifies how many temporary files are to be created. Generally, unless performance issues are experienced, we recommend that this be kept to its default value. Minimum value of `licensestatswqnum` is 1 and maximum is 9999.



Example:

```
licensestatswqnum 1000
```

Default

```
licensestatswqnum 999
```

Used On

All servers

pblicensedblocktimeout

- **Version 9.4.6 and earlier:** `pblicensedblocktimeout` setting not available.
- **Version 10.0.0 and later:** `pblicensedblocktimeout` setting available.

This setting details the maximum delay, in milliseconds, that the database waits to attempt writing to the database. Generally, unless performance issues are experienced, we recommend that this be kept to its default value.



Example:

```
pblicensedblocktimeout 60000
```

Default

```
pblicensedblocktimeout 10000
```

Used On

All servers

pblicensequeuetimeouts

- **Version 9.4.6 and earlier:** `pblicensequeuetimeouts` setting not available.
- **Version 10.0.0 and later:** `pblicensequeuetimeouts` setting available.

This setting details various performance timeout values for use in the license statistics processing. Generally, unless performance issues are experienced, we recommend that this be kept to its default value.



Example:

```
pblicensequeuetimeouts openread=10000,200,1.0 openwrite=10000,200,1.0 write=10000,200,1.0  
lock=10000,200,1.0
```

Default

```
pblicensequeuetimeouts openread=1000,10,1.0 openwrite=10000,10,2.0 write=5000,10,1.0  
lock=30000,10,2.0
```

Used On

All servers



Note: If you experience performance issues, please contact BeyondTrust Technical Support for more details on configuring this setting. For more information, please see www.beyondtrust.com/support.

freelicenseofautoretiredhosts

- **Version 10.3.2 and earlier:** `freelicenseofautoretiredhosts` setting not available
- **Version 21.1.0 and later:** `freelicenseofautoretiredhosts` setting available

The setting `pblicenseretireafter` and the **AutoRetire** attribute of the license string define the number of days after which clients who have not connected are marked as autoretired. A client that is autoretired is available at any time for reuse, thus will continue to consume a license.

Setting `freelicenseofautoretiredhosts` to `no` allows the aforementioned behavior to continue.

Setting `freelicenseofautoretiredhosts` to `yes` causes licenses used by autoretired client hosts to be released by putting them in long-term retirement. Requests coming from such client hosts are not accepted until after the “Recycle” period, which is the minimum number of days after a client is put in long-term retirement. However, this will allow brand-new client hosts to connect and use the freed license slots. This is the default behavior.

The `freelicenseofautoretiredhosts` setting is configured on the Primary License Server.



Example:

```
freelicenseofautoretiredhosts    no
```

Default

```
freelicenseofautoretiredhosts    yes
```

Used On

Primary License Server

REST Services

pbresturi

- **Version 8.5.0 and earlier:** **pbresturi** setting not available.
- **Version 9.0.0 and later:** **pbresturi** setting available.

The **pbresturi** setting is to allow the configuration of non-default REST Uniform Resource Locator paths for the **pblighttpd** service. This setting is primarily for diagnosis. We do not recommend that it be changed in a normal installation.



Example:

```
pbresturi URLSTRING
```

Default

No default value

Used On

All hosts

pbrestlog

- **Version 8.0 and earlier:** **pbrestlog** setting not available.
- **Version 8.1.0 and later:** **pbrestlog** setting available.

The **pbrestlog** setting defines the path to the REST services log.



Example: Depending on the operating system standards, this can be any of the following:

- `/var/log/pbrest.log`
- `/var/adm/pbrest.log`
- `/usr/adm/pbrest.log`

Default

```
pbrestlog /var/log/pbrest.log
```

Used On

All hosts

pbrestdir

- **Version 8.5.0 and earlier:** **pbrestdir** setting not available.
- **Version 9.0.0 and later:** **pbrestdir** setting available.

The **pbrestdir** setting specifies the directory for the installation of the REST services (including **pblighttpd**).



Example:

```
pbrestdir /mypath/rest
```

Default

```
pbrestdir /usr/lib/beyondtrust/pb/<prefix>rest<suffix>
```

Used On

All hosts

pbrestkeyfile

- **Version 8.0 and earlier:** **pbrestkeyfile** setting not available.
- **Version 8.1.0 and later:** **pbrestkeyfile** setting available.

REST services are authenticated using Application ID's and Application Keys. These preshared keys are kept in a database. The **pbrestkeyfile** details the location of this database, and, similarly to all other databases can either be an absolute path, or a path relative to the **basedir** setting.



Example:

```
pbrestkeyfile /mypath/pbrestkeyfile.db
```

Default

```
pbrestkeyfile /opt/<prefix>pbul<suffix>/dbs/pbrstkeys.db
```

Used On

All hosts

pbresttimeout

- **Version 8.0 and earlier:** **pbresttimeout** setting not available.
- **Version 8.1.0 and later:** **pbresttimeout** setting available.

Many internal services in Privilege Management for Unix and Linux are provided by the REST services. The **pbresttimeout** provides a setting to set the maximum amount of time a service will wait until it times out. This timeout is for the overall connection attempt timeout.

For example, if there are 2 log servers with 2 physical addresses for each of them, **pbresttimeout** is the timeout that Privilege Management for Unix and Linux waits to make a successful connection to all 4 physical addresses. That is, each connection waits for 1/4th of **pbresttimeout**.

You can either set a single integer value specifying number of seconds or more detailed format to specify timeouts for specific service. In a detailed format, if connection is not for a specific service or timeout is not specified for a specific connection, default setting is used.

Minimum allowed value is 5 seconds. Maximum allowed value is 86400 seconds (24 hours).



Example: To set a single timeout for all REST services:

```
pbresttimeout 120
```



Example: To set timeout with a detailed format for each service:

```
pbresttimeout default=30 registry=60 pbpolicy=30 logsvr=30 solr=30 logarchive=30  
advkeystrokeactionpolicy=30 fim=30
```

Default

Default value is 30 seconds.

Used On

All hosts

pbrestport

- **Version 8.5.0 and earlier:** **pbrestport** setting not available.
- **Version 9.0.0 and later:** **pbrestport** setting available.

The **pbrestport** setting details the TCP/IP port that REST services use to communicate to remote hosts. This should be consistent across the enterprise installation.

**Example:**

```
pbrestport 3000
```

Default

```
pbrestport 24351
```

Used On

All hosts

pbresttimeskew

REST services use cryptographic methods to communicate, which are relatively time sensitive to provide extra security. We recommend that hosts have their time set precisely or use a suitable protocol implementation such as NTP. If this is not possible, the **pbresttimeskew** details the maximum time difference in seconds between two hosts.

**Example:**

```
pbresttimeskew 120
```

Default

```
pbresttimeskew 60
```

Used On

All hosts

Scheduling Service

schedulingservice

- **Version 9.3.0 and earlier:** `schedulingservice` setting not available.
- **Version 9.4.0 and later:** `schedulingservice` setting available.

The `schedulingservice` option enables the use of the Scheduler, which is required for asynchronous tasks in Privilege Management for Unix and Linux. It is required for Registry Name Service and database synchronization, amongst other things.



Example:

```
schedulingservice no
```

Default

```
schedulingservice yes
```

Used On

All servers

schedulingservicedb

- **Version 10.0.1 and earlier:** `schedulingservicedb` setting not available.
- **Version 10.1.0 and later:** `schedulingservicedb` setting available.

The `schedulingservicedb` option enables the specification of the path to the scheduling service database. It follows similar rules to other database settings in that if it is a relative file it will be appended to the `basedir` setting to get an absolute path.



Example:

```
schedulingservicedb /opt/pbul/dbs/pbsched.db
```

Default

```
schedulingservicedb /opt/<prefix>pbul<suffix>/dbs/pbsched.db
```

Used On

All servers

File Integrity Monitoring

fileintegritydb

- **Version 9.3.0 and earlier:** `fileintegritydb` setting not available.
- **Version 9.4.0 and later:** `fileintegritydb` setting available.

The `fileintegritydb` option specifies the path to the File Integrity Monitoring database. This file is created in `datbasedir` by default, unless the file name starts with '/'.



Example:

```
fileintegritydb /etc/pbfim.db
```

Default

```
fileintegritydb /opt/<prefix>pbul<suffix>/dbs/pbfim.db
```

Used On

All Hosts

fileintegritysignaturesdb

- **Version 9.4.1 and earlier:** `fileintegritysignaturesdb` setting not available.
- **Version 9.4.3 and later:** `fileintegritysignaturesdb` setting available.

The `fileintegritysignaturesdb` option specifies the path to the local clients File Integrity Signature database.



Example:

```
fileintegritysignaturedb /etc/pbfimsigs.db
```

Default

```
fileintegritydb /opt/<prefix>pbul<suffix>/dbs/pbfimsignatures.db
```

Used On

Clients

fileintegrityblocktimeout

- **Version 9.4.4 and earlier:** `fileintegrityblocktimeout` setting not available.
- **Version 9.4.5 and later:** `fileintegrityblocktimeout` setting available.

The `fileintegrityblocktimeout` option specifies in milliseconds the maximum time that the FIM server (`pbconfigd`) waits to acquire a lock on the database. This may be useful when multiple clients send FIM reports at roughly the same time. The lowest useful value is **1000** and the highest is **1200000** (twenty minutes).



Example:

```
fileintegrityblocktimeout 120000
```

Default

```
fileintegrityblocktimeout 10000
```

Used On

FIM Servers

fileintegrityevents

- **Version 9.3.0 and earlier:** `fileintegrityevents` setting not available.
- **Version 9.4.0 and later:** `fileintegrityevents` setting available.

The `fileintegrityevents` option enables the generation of File Integrity Monitoring events for Change Management and Alerting.



Example:

```
fileintegrityevents yes
```

Default

```
fileintegrityevents no
```

Used On

All hosts

Solaris Projects

Solaris 9 introduced the concept of a project, which associates a running process with a Project ID. Administrators can configure resource accounting and resource limitations for the projects.

Solaris administrators can now configure Privilege Management for Unix and Linux secured tasks to be associated with the Solaris project mechanism, and thus take advantage of the project accounting and project resource limits.

Privilege Management for Unix and Linux secured tasks will honor those limitations by not executing when a limit has been reached, and are subject to signals configured for project resource limits.

Privilege Management for Unix and Linux secured tasks can be associated with a Solaris project in one of two ways: with the **PAM pam_setcred()** function or with the **projects.so** library. Privilege Management for Unix and Linux secured tasks are associated with a project that the runuser belongs to.

A Solaris project can be specified on the **pbrun** commandline, or specified in the policy (overrides the commandline), or when not specified, Privilege Management for Unix and Linux 6.1 and 7.0 secured tasks and shells inherit the project from the initiating process (if it has a Solaris project, and the runuser is a member of that project). If the project is not specified and cannot be inherited, the Solaris default project for the runuser is assigned. When a project is not specified, **pbrun** 6.2.6 and 7.0.1+ normally assign the runuser's default project to the secured task. The new keyword **usesubmituserproject** (which defaults to **no**), when set to **yes** will enable the Privilege Management for Unix and Linux 6.1 behavior of inheriting the submituser's project if possible.



Note: The default to **no** changes the default behavior between Privilege Management for Unix and Linux releases. This is a conscious decision to make the "better" behavior the default behavior.

Privilege Management for Unix and Linux shells inherit the project from the previous shell (or assume the default project when used as a login shell).



Note: This behavior can be changed by setting **runsolarisproject** in the policy when (**pbclientmode == "shell start"**). When **iologging** is enabled when **pbclientmode == "shell start"**, the **iologging** parent Privilege Management for Unix and Linux shell runs associated with the user's default project regardless of the setting for **runsolarisproject**.

The project (and runuser) can be changed for subtasks by setting **runsolarisproject** in the policy when (**pbclientmode == "shell command"**). The Privilege Management for Unix and Linux shells require the keyword **enablesolarisprojects** set to **yes** (regardless of the **pamsetcred** setting).

If the Privilege Management for Unix and Linux usage of Solaris projects needs to be disabled on the runhost, set the keyword **pam_setcred()** to **no**, and the keyword **enablesolarisprojects** to **no** on the run host.

Privilege Management for Unix and Linux shells have a safety feature to allow them to operate if Solaris projects are incorrectly configured. If the library specified by the **sharedlibsolarisprojects** keyword cannot be loaded (or is set to none), or if **enablesolarisprojects** is not set to **yes**, Privilege Management for Unix and Linux shell commands function; however, they may operate associated with an incorrect project (the behavior reverts to that in Privilege Management for Unix and Linux v6.0). Errors are logged, but are not displayed to the user.



Note: PAM errors (including Solaris project **pam_setcred()** failures) result in the Privilege Management for Unix and Linux servers shell failing.

sharedlibsolarisprojects

- **Version 6.0 and earlier:** `sharedlibsolarisprojects` setting not available.
- **Version 6.1 and later:** `sharedlibsolarisprojects` setting available.

The `sharedlibsolarisprojects` keyword specifies the location of the Solaris `projects.so` library file on runhosts. The `sharedlibsolarisprojects` keyword is required to function properly whether using the `pamsetcred` keyword, the `enablesolarisprojects` keyword, or both.



Note: This keyword does not apply to `pbssh`. If it is present in the settings file, it does not have any effect on `pbssh` and is ignored.



Example:

```
sharedlibsolarisprojects /usr/lib/libproject.so.1
```

Default

No default value

Used on

Run hosts



For more information, please see the following:

- ["enablesolarisprojects" on page 278](#)
- ["usesubmituserproject" on page 279](#)
- ["pbrun" on page 465](#)

enablesolarisprojects

- **Version 6.0 and earlier:** `enablesolarisprojects` setting not available.
- **Version 6.1 and later:** `enablesolarisprojects` setting available.

On Solaris 9 and 10 runhosts, when Solaris projects are used without using PAM support (`pam`, `pamsessionsservice`, and `pam_setcred`), enable non-PAM support by setting the `enablesolarisprojects` keyword to **yes**. For Solaris 9, PAM is not able to set a project other than the default project. For Solaris 9 runhosts, to allow a project specified on the `pbrun` command line or the Privilege Management for Unix and Linux policy, set the `enablesolarisprojects` keyword to **yes** (regardless of the PAM settings).



Note: This keyword does not apply to **pbssh**. If it is present in the settings file, it does not have any effect on **pbssh** and is ignored.



Example:

```
enablesolarisprojects yes
```

Default

```
enablesolarisprojects no
```

Used on

Run hosts



For more information, please see the following:

- "[pbrun](#)" on page 465
- "[sharedlibsolarisprojects](#)" on page 278
- "[usesubmituserproject](#)" on page 279

usesubmituserproject

- **Version 6.2.5 and earlier, 7.0 and earlier:** **usesubmituserproject** setting not available.
- **Version 6.2.6 and later, 7.0.1 and later:** **usesubmituserproject** setting available.

The **usesubmituserproject** keyword, when set to **yes**, indicates that when a Solaris project is not specified, the secured task is associated with the submituser's current project if possible. When set to **no** and a Solaris project is not specified, or if inheriting the submituser's project is not possible, the secured task is associated with the runuser's default project. This keyword is only effective on Solaris run hosts.



Note: This keyword does not apply to **pbssh**. If it is present in the settings file, it does not have any effect on **pbssh** and is ignored.



Example:

```
usesubmituserproject yes
```

Default

```
usesubmituserproject no
```

Used on

Run hosts



For more information, please see the following:

- *"enablesolarisprojects" on page 278*
- *"sharedlibsolarisprojects" on page 278*
- *"pbrun" on page 465*
- *"pblockd" on page 430*

Daemon Mode

daemonfork

- **Version 9.3.0 and earlier:** `daemonfork` setting not available.
- **Version 9.4.0 and later:** `daemonfork` setting available.

The `daemonfork` keyword controls whether Privilege Management daemons make a second fork when run in daemon mode.

When the `daemonfork` keyword is set to **yes**, Privilege Management daemons (in daemon mode) fork after the `setsid()` call, meaning that they are no longer process group leaders or session leaders (for example, an *old-style* true *daemon*). Some init mechanisms (for example, `upstart`) lose track of the daemon's pid when this is set to **yes**.



Example:

```
daemonfork yes
```

Default

```
daemonfork no
```

Used On

- Policy servers
- Log servers
- Run hosts

pidfilepath

- **Version 9.3.0 and earlier:** `pidfilepath` setting not available.
- **Version 9.4.0 and later:** `pidfilepath` setting available.

`pidfilepath`, when specified, is the path for Privilege Management daemon pid files, named `<pidfilepath>/<prefix><daemonname><suffix>.pid`. when run in daemon mode and not foreground mode, the `pbmasterd`, `pblocald`, `pblogd`, and `pbsyncd` daemons write the pid of the daemon after any forks.



Example:

```
pidfilepath /var/run/
```

Default

none

Used On

- Policy servers
- Log servers
- Run hosts

Reporting

replaytimeformat

- **Version 9.3.0 and earlier:** `replaytimeformat` setting not available.
- **Version 9.4.0 and later:** `replaytimeformat` setting available.

The `pbreplay --timestamp (-t)` option that can be used to specify the time format, using the format sequences that the `date` command uses.

Starting with version 9.4, the `replaytimeformat` keyword in `pb.settings` can be used to permanently specify a time format. The `commandline` option overrides the keyword. If the keyword is not specified, behavior is the same as pre v.9.4.

`pbinstall` creates the `replaytimeformat` with the default value `%a %b %d %Y %r`, resulting in date/time displayed in weekday month day year 12 hour AM/PM format.



Example:

```
Tue Aug 09 2016 04:52:44 PM
```

Default

`%a %b %d %Y %r` (via `pbinstall`)

Used On

- Policy servers
- Log servers



For a list of format sequences, please see *man date*.

Manage and Test System Configuration

Privilege Management for Unix and Linux system administration sometimes involves validating or troubleshooting an installation. These tasks are performed using the Privilege Management for Unix and Linux **pbbench** and **pbcheck** utilities.

pbbench

The **pbbench** program provides useful information for solving configuration, file permission, and network problems. It reads the Privilege Management for Unix and Linux settings file on the local machine and uses system information, such as that found in **/etc/services** and **/etc/hosts** or NIS, to verify the information in the settings file.

If **pbbench** detects an error, then it displays an error message. Output consists of information about the tests that were performed, the results of the tests, and any errors that were encountered. By default, this output goes to standard error. If no errors are detected, then **pbbench** returns silently.

pbbench checks for very old versions of Privilege Management for Unix and Linux (pre-2.0) by looking for **/etc/pb.ports** and **/etc/pb.masters** and reports a warning if these are found. The HTML GUI version of **pbbench** does not check for the Privilege Management for Unix and Linux pre-v2 files.

pbbench is treated as a user program and can be run by root and non-root users. However, some non-root user queries might fail for lack of permissions. The location of **pbbench** can be set during Privilege Management for Unix and Linux installation. The default location of **pbbench** is **/usr/local/bin**.

To run **pbbench** and redirect the output to a file rather than to standard error, use the command:

```
pbbench > filename
```



Example:

```
pbbench > pbbench.output
```

Privilege Management for Unix and Linux expects to find some file permissions and network configurations in a certain condition. Privilege Management for Unix and Linux might or might not run depending on these conditions. **pbbench** generates **INFO**, **WARNING**, and **ERROR** messages to report its findings. Some of the findings are merely informational, but some need to be heeded.



For more information, please see "[pbbench](#)" on page 366.

pbcheck

The Privilege Management for Unix and Linux **pbcheck** utility provides the capability to test the policy file and also to produce data that describes which commands can run under what conditions.



For detailed information on this utility, please see "[pbcheck](#)" on page 371.

Testing Policies

pbcheck can perform syntax, type, and other checks on a policy file. A common use for this utility is to test a new policy for errors before installing it on a live system. To test a new policy, enter the following:

```
pbcheck -f filename
```

or

```
pbcheck --file= filename
```

Another typical use for **pbcheck** is to check the syntax in an existing policy file without executing a function or procedure. This action is done with the following command:

```
pbcheck -s
```

or

```
pbcheck --syntax
```

With no options, **pbcheck** performs a run-check on the configuration policy file that is specified in the settings file.

Entitlement Reporting

- **Version 4.0 and earlier:** entitlement reporting not available.
- **Version 5.0 and later:** entitlement reporting available.

Entitlement reporting is an essential element of audit control (for Sarbanes-Oxley compliance in the U.S., for example). Beginning with Privilege Management for Unix and Linux v5.0, **pbcheck** can use the Privilege Management for Unix and Linux parser to emulate simple policies and produce data that describes which commands can run under what conditions.

The resulting data is presented as comma-separated values that can be fed to the Privilege Management for Unix and Linux report writer to produce a full entitlement report, or be exported to other programs.

Because the Privilege Management for Unix and Linux policy language is so extensive, entitlement reporting has two important limitations:

1. Policies can use external data sources and programs. Privilege Management for Unix and Linux can base entitlement decisions on these external sources. However, these external sources can produce different results without the policy ever changing.
2. Policies can be quite complex. As a result, a complex policy could produce an incomplete report.



For more information about additional limitations of entitlement reporting, please see "pbcheck" on page 371.

License Management

Each time a user submits a request using **pbrun** or when an administrator runs the File Integrity Monitor, there is a check with the license services that a valid license is present. Without a valid license, Privilege Management for Unix and Linux does not accept requests from users.



Note: Introduced in version 10.0, a license string consists of a JSON (JavaScript Object Notation) string that details expiry, facilities, and services.

There are two types of licenses:

- **Temporary:** The **HostId** attribute is set to **temporary** and is installed automatically to allow customers to evaluate Privilege Management for Unix and Linux.
- **Standard:** Supplied by BeyondTrust once a customer has purchased the product. A standard license has a **HostId** attribute that associates the license with the primary license server of the Privilege Management for Unix and Linux installation.



Example: Temporary License String:

```
{ "PBULPolClnts":20, "SudoPolClnts":20, "RBPClnts":20, "ACAClnts":1, "AKAClnts":20,
  "FIMClnts":20, "SOLRClnts":1, "Owner":"Temporary License", "Comment":"Temporary License",
  "AutoRetire":7, "Recycle":7, "Expires":"2018-03-11 00:00:00", "Terminates":"2018-04-10
  00:00:00", "HostId":"temporary", "HMAC":"UtGE3tD6qK2UwutY3GFOqodjdq30pEDAW2cKb5/OaMc=" }
```

A temporary license is installed automatically if a standard license is not provided when the primary license server is installed. It enables 20 client seats for all services and enables all facilities. The license is valid for 60 days.



Note: When requesting a standard license, you are asked to provide the output of **pbadm** **--info --uuid** from the host that will run the primary license service. This displays the UUID (Universal Unique Identifier) that identifies the host. From this, BeyondTrust can generate a license that is associated directly to the host, with the appropriate facilities and services. This can then be imported into the primary license server.



Example: pbadm --info --uuid Output:

```
7faf7681-4d42-4b69-00bf-dad93b4a3dfb
```



Example: Standard License String:

```
{ "PBULPolClnts":200, "SudoPolClnts":200, "RBPClnts":200, "ACAClnts":1, "AKAClnts":0,
  "FIMClnts":0, "SOLRClnts":1, "Owner":"My Company Corp", "Comment":"Standard License for
  My Company", "AutoRetire":7, "Recycle":7, "Expires":"2018-03-01 00:00:00",
  "Terminates":"2019-03-01 00:00:00", "HostId":"7faf7681-4d42-4b69-00bf-dad93b4a3dfb",
  "HMAC":"UtGE3tD6qK2UwutY3GFOqodjdq30pEDAW2cKb5/OaMc=" }
```

The license string introduced in version 10.0 of Privilege Management for Unix and Linux consists of a list of attributes that are human-readable and detail the entitlement of the license. These attributes are:

RBPCInts: <num>	Details the maximum number of clients that are licensed to use Privilege Management for Unix and Linux role-based policy. A value of 0 means there is no entitlement. A value of -1 means unlimited clients.
AKACInts: <num>	Details the maximum number of clients that are licensed to use Privilege Management for Unix and Linux Advanced Keystroke Action functionality. A value of 0 means there is no entitlement. A value of -1 means unlimited clients.
FIMCInts: <num>	Details the maximum number of clients that are licensed to use Privilege Management for Unix and Linux File Integrity Monitor. A value of 0 means there is no entitlement. A value of -1 means unlimited clients.
ACACInts: <0 1>	Details whether the license allows the use of Advanced Control and Audit.
SOLRCInts: <0 1>	Details whether the license allows the use of SOLR Indexing functionality.
Owner: <name>	Details the owner of the license.
Comment: <string>	A simple string that can be updated to include any further information you wish to include.
AutoRetire: <num>	Details the minimum duration in days after which a license is automatically retired due to client inactivity, allowing the license to be used by another client.
Recycle: <num>	Details the minimum number of days after a client has been manually retired before it can be used again.
Expires: <date>	Details when the license runs out, after which messages are displayed within the log files, and eventually to the end users, to remind the administrator to renew the license. The product continues to run without otherwise affecting functionality.
Terminates: <date>	The cut off date for the product, after which it ceases to function.
HostId: <string>	Details the host UUID designated to the primary license server.
HMAC: <string>	Provides security to the license and customer to ensure that the license is authentic and correct, and has not been corrupted or altered.

The license server hosts manage licensing information by storing the client's host ID in the Privilege Management centralized license database to keep track of client connections. Licensing management is provided by the **pbadmin** tool, using the **--lic** options. The first installation of the license server is the primary license server. This installs a temporary license if no standard license is provided. Any machine that runs a client component consumes a Privilege Management for Unix and Linux license, even if the machine is also a policy server host or log host.

Beginning with version 7.1, Privilege Management for Unix and Linux can optionally track the last access date and node name of the clients.

Prior to version 8.5, policy servers used the connecting client's IP address as the identifier to differentiate it from other client hosts. However, IP addresses could not always assure client host uniqueness.

For instance, a host with multiple network interface cards (NICs) can have multiple IP addresses. The IP address of a client host that previously connected to a policy server could be changed.

Starting with version 8.5, Privilege Management for Unix and Linux uses UUIDs (universally unique identifiers) instead of IP addresses to identify and track connected clients. The UUID is derived from operating system calls and is unique to that host.

Starting with version 10.0, licenses are synchronized across all servers in the Privilege Management enterprise installation. This makes licensing administration easier by giving a single pool of licenses available to the whole application, which can be managed from a single host.

Client Limit Enforcement

Starting with version 8.5, a policy server softens enforcement of its license limit to give customers time to contact BeyondTrust and adjust their license.

Privilege Management for Unix and Linux clients up to 10% over the license client limit are allowed to connect to the policy server without producing any error.

If the policy server receives client connections between 11%-20% over the license limit, a warning is written in syslog and the policy server diagnostic log file. If the policy server receives client connections between 21%-50% over the license limit, the initiating client program also receives a warning message. When a new client connection exceeds the license limit by more than 50%, an error displays, and the policy server rejects any new connection requests at that point.

Command Line Management

Licenses are managed on the primary license server using the program **pbadmin**. Only the root user can run **pbadmin**.

Usage

```
pbadmin --lic [<options>] [ <file> <file>...]
```

-u '{ param }'	Update primary license server license where the { param } argument is the supplied JSON formatted license.
-u <path> --force	Update primary license server license from file. Force the license update on secondary license server.
-G	Retrieve license string and attributes.
-l [<wildcard(s)>]	List client license usage summary.
-l	Add an extra -l to list client usage detail.
-l [{ ["fqdn" : "<wildcard>"], ["retired" : <true false>], ["updated_older" : <epoch>],["updated_newer" : <epoch>], ["updated_older" : { "years" : n, "months" : n, "days" : n, "hours" : n } ["updated_newer" : { "years" : n, "months" : n, "days" : n, "hours" : n } }]	List clients with attributes.
-s <[- +]attribute>	Sort the list of records by attribute name (ascending/descending).

-L [<service>]	List Client Service License Usage Summary.
-L	Add an extra -L to list all client service details.
-r {"uuid" : "<uuid wildcard>"} -r {"uuid" : ["<uuid wildcard>", "uuid", ...]} -r {"fqdn" : "<fqdn wildcard>"} -r {"fqdn" : ["<fqdn wildcard>", "fqdn", ...]}	Retire clients to free up licenses based upon UUID, FQDN, or wildcard.
-R	Refresh license statistics from the primary license server.

License Management

pbadmin is used to import standard licenses using the **-u** option. It can be used to list client and service statistics, and it can be used to retire old client licenses.

As of version 10.0, License Management is centralized and can be carried out on the primary license server using the command **pbadmin**.

The command line administration tool provides methods to update the license string, to list summary statistics and to retire clients to free up licenses.

All of the commands that list statistics can be run from any server that provides a service. All commands that update the database, such as updating the license itself or retiring clients, should be run on the primary license server:



Example:

```
pbadmin --lic -u '{ "PBULPolClnts":200, "SudoPolClnts":200, "RBPClnts":200, "ACAClnts":1,
"AKAClnts":0, "FIMClnts":0, "SOLRClnts":1, "Owner":"My Company Corp", "Comment":"Standard
License for My Company", "AutoRetire":7, "Recycle":7, "Expires":"2018-03-01 00:00:00",
"Terminates":"2019-03-01 00:00:00", "HostId":"7faf7681-4d42-4b69-00bfdad93b4a3dfb",
"HMAC":"UtGE3tD6qK2UwutY3GFOqodjdq30pEDAW2cKb5/OaMc="} '
```

This command updates the installation with the license string provided by BeyondTrust to a standard license.

Listing Clients

To view a list of client hosts that are using the license on a license server, use the **pbadmin --lic -l** command option. Doing so produces a list similar to the one in the following example:



Example:

```
# pbadmin -P --lic -l
{
  "uuid": "7faf7681-4d42-4b69-00bf-dad93b4a3dfb",
  "fqdn": "pbuild",
```



```

"addr": "[{"family":4,"port":0,"addr":"192.168.16.138"}]",
"lastupdated": "2018-01-17 09:31:37",
"retired": "never",
"recycle": "never"
}
{
"uuid": "e5368b17-e78b-416c-ae1a-14c6273cccb6",
"fqdn": "host2",
"addr": "[{"family":4,"port":0,"addr":"192.168.16.140"}]",
"lastupdated": "2018-01-17 09:31:40",
"retired": "2018-01-17 09:31:40",
"recycle": "never"
}
{
"uuid": "f5d5edd2-2ed0-4fe3-bbb5-46e6ce295b22",
"fqdn": "host3",
"addr": "[{"family":4,"port":0,"addr":"192.168.16.141"}]",
"lastupdated": "2018-01-17 09:26:40",
"retired": "never",
"recycle": "never"
}
}
    
```

Note the retired license in the list.

Retiring Clients

If a client is removed from the network, then its slot in the license database remains active. To retire a client host from the license database, specify the **-r** command option for **pbadmin**.



Example: To retire one client:

```

# pbadmin --lic -r '{ "uuid" : "2ddf83e6-aabf-4dbe-a70e-f73a3d73aea6" }'
*** WARNING ***
You have elected to retire clients from.
Clients which are retires will be usable to reconnect to the application
For a period of 7 days, regardless of the number of unused licenses remaining.
If you are sure you want to do this, please specify the '--force' option
# pbadmin --lic -r '{ "uuid" : "2ddf83e6-aabf-4dbe-a70e-f73a3d73aea6" }' --force
    
```



Example: To retire multiple clients:

```

# pbadmin --lic -r '{ "uuid" : ["2ddf83e6-aabf-4dbe-a70e-f73a3d73aea6","..."]} ' --force
    
```

*This example demonstrates the retirement of a client host with **2ddf83e6-aabf-4dbe-a70e-f73a3d73aea6** UUID from the active licenses. This action frees up one license slot for another.*

License Settings

i For the list and description of settings related to license management, please see "[Licensing](#)" on page 264.

Registry Name Service and Database Synchronization

Privilege Management for Unix and Linux has historically required manual or customer-developed processes to copy policy from one policy server to another to make sure that failover and load balancing were addressed. Privilege Management for Unix and Linux has historically required manual or customer-developed processes to copy encryption key files from one host to another to ensure no interruption in communication between its components. Hostnames or IP addresses were hard-coded into the settings file on every client and server to detail which servers provided services for each client. Although the use of DNS could mitigate some of these issues, day-to-day maintenance of these settings and policy configuration across multiple hosts remains arduous and open to user error.

As a solution to these problems, a flexible and powerful Registry Name Service incorporating Database Synchronization has been developed and integrated into Privilege Management to provide advanced failover and load balancing automatically, centralized role based management, and the ability to form groups of clients that share configuration or policy based on role or business organization.

Service Groups and Planning

The Registry Name Service provides the product with a method of addressing and locating other components of the BeyondInsight product within the enterprise. Each category of service, including Privilege Management for Unix and Linux Policy Authorization, Logging Services, and the Registry Name Service itself, have distinct groups which are comprised of a single primary server and zero or more secondary servers.

The primary host accepts all the configuration changes within that specified Service Group and automatically synchronizes these changes to the secondary service hosts. Other functions, such as authorization or data retrieval, are available from any of the secondary hosts within the Service Group, providing effective load balancing for clients.

Each host that makes up the Service Group is defined in the Registry Name Service database, including primary and secondary servers and clients. This allows every host within the BeyondInsight enterprise to identify every machine that will make up its Service Group. This also allows a more fine-grained control of licenses within the product.

When upgrading or installing a new BeyondInsight enterprise and incorporating the Registry Name Service, the first server installed becomes the Primary Registry Name Server. This creates a repository with the default groups of **registry_name_service**, **dflt_pbpolicy_service**, **dflt_log_service**, **dflt_sudopolicy_service**, **dflt_logarch_service**, and **dflt_fim_service**.

Once the primary Registry Name Server has been installed, the administrator is free to define more Service Groups to allow multiple areas of the enterprise to work and be administered autonomously. Then, for each Service Group, the administrator can start installing the primary servers, secondary servers, and finally the clients. As soon as a host is added to a Service Group and its role within the group is defined, it will be available to use. Secondary servers are provided with configuration from the primary, and clients are aware of service providers immediately.



Note: There can only be one Registry Name Service group.

Hosts are members of one instance of each type of Service Group that is available. For example, a typical BeyondInsight policy client is a member of a single BeyondInsight Policy Service Group and a single Privilege Management Log Services Group. Within each of these groups the primary and secondary servers provide the load balancing and failover required by mission-critical environments.

Command Line Configuration

The configuration of the Registry Name Service is provided through the enhanced **pbdbutil** command line utility. New categories have been added.

--svc is used on the Primary Registry Name Server to configure the Registry Name Service Groups:

```
Usage pbdbutil --svc [<options>] [ <file> <file> ...]
```

<code>-u '{"svcgname" : "name", params...}'</code>	Create/Update Registry Name Service Group.
<code>-u '{"cn" : "hname", params...}'</code>	Create/Update Registry Name Service Host.
<code>-u '{"cn" : "hname", "uuid" : "", params...}'</code>	Create/Update external Host in Registry Name Service.
<code>-u '{"svcgname" : "name", "cn" : "hname", params...}'</code>	Add/Update Registry Name Service Host to Service Group.
<code>-g '{"svcgname" : "name"}</code>	Retrieve Registry Name Service Group information.
<code>-g '{"primary" : "name"}</code>	Lookup the Primary Server within the Registry Name Service Group.
<code>-g '{"cn" : "name"}</code>	Retrieve Registry Name Service Host information by host common name.
<code>-g '{"uuid" : "name"}</code>	Retrieve Registry Name Service Host information by UUID.
<code>-d '{"svcgname" : "name"}</code>	Delete Registry Name Service Group.
<code>-d '{"svcgname" : "name", "cn" : "name"}</code>	Remove a host from Registry Name Service Group.
<code>-d '{"cn" : "name"}</code>	Delete Registry Name Service Host by host common name.
<code>-z <oldgrp> <newgrp></code>	Rename Registry Name Service Group.
<code>-l [<wildcard(s)>]</code>	List all the Registry Name Service Groups that match wildcards.
<code>-l</code>	Add an extra <code>-l</code> to list Servers in the Registry Name Service Groups.
<code>-l</code>	Add a third <code>-l</code> to list all hosts in the Registry Name Service Groups.
<code>-L [<wildcard(s)>]</code>	List all the Hosts that match wildcards.
<code>-L</code>	Add an extra <code>-L</code> to list Service Group membership and role.
<code>-p <svcgrp> <host></code>	Promote host to Primary Service within the specified Registry Name Service Group.
<code>-N [[<cn> [<port>]]</code>	Create and initialize Primary Registry Name Service database.
<code>-n</code>	Create new Registry Name Service database.

--scache is used on all hosts that are subscribed to the Registry Name Service and enables the interrogation of the local host Registry Name Service cache:

```
Usage pbdbutil --scache [<options>] [ <file> <file> ...]
```

<code>--cn</code>	Retrieve Common Name from the Registry Name Service.
<code>-w</code>	Retrieve my Registry Name Service information.

-l	List all the locally cached Registry Name Service entries.
-s <[- +]attribute>	Sort the list of records by attribute name (ascending/descending).
-R	Refresh the local Registry Name Service cache.
-N { param }	<p>Create and initialize the Primary Registry Name cache database where the { param } argument is formatted JSON with parameters:</p> <ul style="list-style-type: none"> • "hostname" : "host1": Hostname of the Registry Manager REST service. • "port" : 24351: Port of the Registry Manager REST service. • "appid" : "appid": App ID of the Registry Manager REST service. • "appkey" : "xxx-xxx-xxx": App key of the Registry Manager REST service.
-m <msg>	Specify message (required when change management enabled).

--dbsync provides information regarding the status of Database Synchronization on primary servers:

```
Usage pbdbutil --dbsync [<options>] [ <file> <file> ...]
```

-l	List Database Synchronization history.
-l [<dbfile(s)>]	List outstanding Database Synchronization entries.
-c <dbfile(s)>	Clear the outstanding synchronization entries from database.
-R <svc> [<cn>]	Resynchronize database for specified service. Starting with v10.3.0, issuing a pbdbutil --dbsync -R initiates a database synchronization immediately, instead of waiting for the next dbsyncrefresh time (as it was done prior to 10.3.0).
-A <svcname> <...>	Set databases in Service Group(s) as being automatically synchronized.
-X <svcname> <...>	Unset databases in Service Group(s) as being automatically synchronized.

Settings and Configuration

Prerequisites

The BeyondInsight install process configures individual hosts appropriately to use the Registry Name Service from the outset. However, if BeyondInsight is upgraded or configured manually to use the Registry Name Service there are a number of settings and commands that need to be run to successfully utilize the service.

pb.settings:

"registrynameservice yes"	Required in every host that utilizes Registry Name Service.
submitmasters, acceptmasters, logservers	To look up servers in the Registry Name Service a single asterisk is used. Each of these settings can be set and migrated individually and can be used with hostnames or IP addresses appended if hard-coded failover servers are desired.
servicedb	This is required on Primary and Secondary Name Servers to specify the path to the Registry Name Service database.
svccachedb	This is required on all hosts to specify the path to the Registry Name Service Cache database.
dbsyncdb	This is required on all primary hosts to specify the path to the Database Synchronization database.

We recommend that you apply these settings initially to the Primary Registry Name Server and then, as hosts are added, into the Registry Name Service.

registrynameservice

- **Version 9.3.0 and earlier:** **registrynameservice** setting not available.
- **Version 9.4.0 and later:** **registrynameservice** setting available.

The **registrynameservice** option provides a global switch on each host to turn Registry Name Services on or off. Once it is turned on, individual settings such as **submitmaster**, **acceptmaster**, and **logservers** must be configured with a single asterisk to enable each setting to look up information in the Registry Name Service.



Example:

```
registrynameservice yes
```

Default

```
registrynameservice no
```

Used On

All hosts

rnsoptions

- **Version 10.2.0 and earlier:** **rnsoptions** setting not available.
- **Version 10.3.0 and later:** **rnsoptions** settings available.

```
rnsoptions [UseFQDN|UseAllIPs]
```

If set to **UseFQDN**, RNS uses only the first IP address in the RNS address list to contact a host.

When set to **UseAllIPs**, it uses all IP addresses held within RNS to contact a host.

These are mutually exclusive.



Example:

```
rnsoptions UseAllIPs
```

Default

No default value

Used On

All RNS hosts

servicedb

- **Version 9.3.0 and earlier:** **servicedb** setting not available.
- **Version 9.4.0 and later:** **servicedb** setting available.

The **servicedb** option specifies the path to the Registry Name Service Database. This file is created in **basedir** by default, unless the file name starts with a slash (/).



Example:

```
servicedb /etc/pbsvc.db
```


Default

```
servicedb /opt/<prefix>pbul<suffix>/dbs/pbsvc.db
```

Used On

Registry Name Server

svccachedb

- **Version 9.3.0 and earlier:** **svccachedb** setting not available.
- **Version 9.4.0 and later:** **svccachedb** setting available.

The **svccachedb** option defines the path to the Service Cache Database. This file is created in **databasedir** by default, unless the file name starts with a slash (/).



Example:

```
svccachedb /etc/svccache.db
```

Default

```
svccachedb /opt/<prefix>pbul<suffix>/dbs/pbsvccache.db
```

Used On

All hosts, when Registry Name Service is enabled.

svccacherefresh

- **Version 9.3.0 and earlier:** **svccacherefresh** setting not available.
- **Version 9.4.0 and later:** **svccacherefresh** setting available.

The **svccacherefresh** option defines how often the Registry Name Service Cache Database is checked against the Registry Name Server for updates. Smaller values allow clients to retrieve changes in configuration in the Registry Name Service more quickly, but produce more network and load on the Registry Name Servers.



Example:

```
svccacherefresh 120
```

Default

```
svccacherefresh 110
```

Used On

All hosts, when Registry Name Service is enabled.

warnusersvccache

- **Version 10.2.0 and earlier:** `warnusersvccache` setting not available.
- **Version 10.3.0 and later:** `warnusersvccache` setting available.

The `warnusersvccache` option displays RNS Service Cache out of date message to `pbrun` user.



Example:

```
warnusersvccache yes
```

Default

```
warnusersvccache no
```

Used On

All hosts, when Registry Name Service is enabled.

Primary Registry Name Server Configuration

To create and initialize the Registry Name Service on the Primary Registry Name Server, use:

```
# pbdbutil --svc -N --force
```

Several items are created:

- The database
- The default Service Groups
- A host record for the primary server, with:
 - The appropriate Common Name set to the local hostname
 - A Fully Qualified Domain Name
 - A role configured as Primary Registry Name Server in the Registry Name Service Group.

This can be checked using:

```
# pbdbutil -P --svc -l -l
{
  "svcgid": 1,
  "svcgname": "registry_name_service",
  "svc": "registry",
  "updated_usec": "2016-11-10 11:12:20",
  "deleted": false,
  "svcs": [
    {
      "svcgid": 1,
      "hostid": 1,
      "role": "primary",
      "sorder": 1,
      "created_usec": "2016-11-10 11:12:20",
      "updated_usec": "2016-11-10 11:12:20",
      "cn": "pbulprimrns",
      "uuid": "3d13a9eb-7340-4199-aa47-1570941bd50f",
      "fqdn": "pbulprimrns.org.com",
      "addrs": [
        {
          "family": 4,
          "port": 24351,
          "addr": "192.168.1.1"
        }
      ],
      "tnlzone": 0,
      "deleted": 0
    }
  ]
}
{
  "svcgid": 2,
  "svcgname": "dflt_pbpolicy_service",
  "svc": "pbpolicy",
  "updated_usec": "2016-11-10 11:12:20",
  "deleted": false
}
{
  "svcgid": 3,
  "svcgname": "dflt_log_service",
  "svc": "logsvr",
  "updated_usec": "2016-11-10 11:12:20",
  "deleted": false
}
{
  "svcgid": 4,
  "svcgname": "dflt_sudopolicy_service",
  "svc": "sudopolicy",
  "updated_usec": "2016-11-10 11:12:20",
  "deleted": false
}
{
  "svcgid": 5,
  "svcgname": "dflt_Solr_service",
```

```
"svc": "Solr",
"updated_usec": "2016-11-10 11:12:20",
"deleted": false
}
{
"svcgid": 6,
"svcgname": "dflt_logarch_service",
"svc": "logarchive",
"updated_usec": "2016-11-10 11:12:20",
"deleted": false
}
{
"svcgid": 7,
"svcgname": "dflt_beyondinsight_service",
"svc": "beyondinsight",
"updated_usec": "2016-11-10 11:12:20",
"deleted": false
}
{
"svcgid": 8,
"svcgname": "dflt_fim_service",
"svc": "fim",
"updated_usec": "2016-11-10 11:12:20",
"deleted": false
}
}
```

Please note the use of **-P** to print the output in a pretty format to make it easier to read.

Add Further Hosts Into the Enterprise

Further hosts can be added to the Registry Name Service in two ways. New hosts can be added on installation by using the Client Registration option in **pbinstall**. If this is selected and a suitable Client Registration profile is used, detailing default Registry Name Service Groups, the host is automatically added to the default Service Groups as a client, depending upon the host function selected at install time.

However, if automatic registration is not used, the host can be manually added to the Registry Name Service.

First, the host's unique UUID is required. On the host run:

```
# pbdbutil --info --uuid 969ecab2-93d8-4322-a8cf-6314457053bf
```

Then use this to add the host on the Primary Registry Name Server:

```
# pbdbutil --svc -u '{"cn":"pbtest","fqdn":"pbtest.org.com","uuid":"969ecab2-93d8-4322-a8cf-6314457053bf"}'
```

The Fully Qualified Domain Name (FQDN) is used to look up the host's address in the local Name Service. If the FQDN is not supplied, the Common Name (CN) is used instead.

Once the host has been added, it can be added to the specified Service Group as a particular role:

```
# pbdbutil --svc -u '{ "svcgname" : "test_pbpolicy", "cn" : "pbtest", "role" : "client" }'
```

If the host is added as a secondary server to a Service Group that already has a primary server, it starts receiving configuration automatically from the database synchronization. The license database is synchronized on the server when the role changes from client to primary license server.

Routine Configuration Examples

A list of hosts contained with the Registry Name Service is retrieved using:

```
# pbdbutil -P --svc -L
{
  "hostid": 1,
  "cn": "pbulprimrns",
  "uuid": "3d13a9eb-7340-4199-aa47-1570941bd50f",
  "fqdn": "pbulprimrns.org.com",
  "addrs": [
    {
      "family": 4,
      "port": 24351,
      "addr": "192.168.1.1"
    }
  ],
  "tnlzone": 0,
  "updated_usec": "2016-11-10 11:12:20",
  "deleted": false
}
```

Add New Service Groups

```
# pbdbutil --svc -u '{ "svcgname" : "test_pbpolicy", "svc" : "pbpolicy" }'
# pbdbutil --svc -l
{"svcgid":1,"svcgname":"registry_name_service","svc":"registry","updated_usec":"2016-11-10
11:12:20","deleted":false}
{"svcgid":2,"svcgname":"dflt_pbpolicy_service","svc":"pbpolicy","updated_usec":"2016-11-10
11:12:20","deleted":false}
{"svcgid":3,"svcgname":"dflt_log_service","svc":"logsvr","updated_usec":"2016-11-10
11:12:20","deleted":false}
{"svcgid":4,"svcgname":"dflt_sudopolicy_service","svc":"sudopolicy","updated_usec":"2016-11-10
11:12:20","deleted":false}
{"svcgid":5,"svcgname":"dflt_Solr_service","svc":"Solr","updated_usec":"2016-11-10
11:12:20","deleted":false}
{"svcgid":6,"svcgname":"dflt_logarch_service","svc":"logarchive","updated_usec":"2016-11-10
11:12:20","deleted":false}
{"svcgid":7,"svcgname":"dflt_beyondinsight_service","svc":"beyondinsight","updated_usec":"2016-
11-10 11:12:20","deleted":false}
{"svcgid":8,"svcgname":"dflt_fim_service","svc":"fim","updated_usec":"2016-11-10
11:12:20","deleted":false}
{"svcgid":100,"svcgname":"test_pbpolicy","svc":"pbpolicy","updated_usec":"2016-11-10
11:32:42","deleted":false}
```

The default groups have Service Group IDs less than 100 and cannot be removed.

Retrieve Specified Service Group Information

```
# pbdbutil --svc -g '{ "svcgname" : "test_pbpolicy" }'  
{ "svcgid":100,"svcgname":"test_pbpolicy","svc":"pbpolicy","updated_usec":"2016-11-10  
11:32:42","deleted":false}
```

Retrieve Specified Host by Common Name

```
# pbdbutil --svc -g '{ "cn" : "pbulprimrns" }'  
{ "cn":"pbulprimrns","uuid":"3d13a9eb-7340-4199-aa47-  
1570941bd50f","fqdn":"pbulprimrns.org.com","adrs":  
[{"family":4,"addr":"192.168.1.1","port":24351}]}
```

Retrieve Specified Host UUID

```
# pbdbutil --svc -g '{ "uuid" : "3d13a9eb-7340-4199-aa47-1570941bd50f" }'  
{ "cn":"pbulprimrns","uuid":"3d13a9eb-7340-4199-aa47-1570941bd50f",  
"fqdn":"pbulprimrns.org.com","adrs":[{"family":4,"addr":"192.168.1.1","port":24351}]}
```

Retrieve the Primary Server for the Specified Service Group

```
# pbdbutil --svc -g '{ "primary" : "registry_name_service" }'  
{ "svcgid":1,"svcgname":"registry_name_service","svc":"registry","updated_usec":"2016-11-10  
11:12:20","deleted":false,"hostid":1,"role":"primary","sorder":1,"created_usec":"2016-11-10  
11:12:20","cn":"pbulprimrns","uuid":"3d13a9eb-7340-4199-aa47-  
1570941bd50f","fqdn":"pbulprimrns.org.com","adrs":  
[{"family":4,"port":24351,"addr":"192.168.1.1"}],"tnlzone":0}
```

Retrieve the Current Hosts Information from the Registry Name Service Cache

```
pbdbutil --scache -w  
{ "fqdn":"pbulprimrns.org.com","cn":"pbulprimrns","uuid":"969ecab2-93d8-4322-a8cf-  
6314457053bb","adrs":[{"addr":"192.168.16.138","family":4,"port":24351}]}
```

Retrieve the Complete List of Service Groups and Hosts

```
# pbdbutil -P --svc -l -l -l  
{  
  "svcgid": 1,  
  "svcgname": "registry_name_svc",  
  "svc": "registry",  
  "updated": "2016-06-14 10:43:14",  
  "deleted": 0,
```

```
"svcs": [
  {
    "svcgid": 1,
    "hostid": 1,
    "role": "primary",
    "created": "2016-06-14 10:43:14",
    "updated": "2016-06-14 09:43:14",
    "deleted": 0,
    "cn": "pbulprimrns",
    "uuid": "969ecab2-93d8-4322-a8cf-6314457053bb",
    "fqdn": "pbulprimrns.org.com",
    "addrs": [
      {
        "family": 4,
        "port": 24351,
        "addr": "192.168.1.1"
      }
    ],
    "tnlzone": 0
  }
]
.
.
.
{
  "svcgid": 100,
  "svcgname": "test_pbpolicy",
  "svc": "pbpolicy",
  "updated": "2016-06-14 09:52:17",
  "deleted": 0,
  "svcs": [
    {
      "svcgid": 100,
      "hostid": 4,
      "role": "client",
      "created": "2016-06-14 11:06:46",
      "updated": "2016-06-14 10:05:03",
      "deleted": 0,
      "cn": "pbtest",
      "uuid": "969ecab2-93d8-4322-a8cf-6314457053bf",
      "fqdn": "pbtest",
      "addrs": [
        {
          "family": 4,
          "port": 24351,
          "addr": "192.168.1.5"
        }
      ],
      "tnlzone": 0
    }
  ]
}
```

Delete the Host

```
# pbdbutil --svc -d '{ "cn" : "pbtest" }'
```

Add the New Host as a Primary Server

```
# pbdbutil --svc -u '{ "svcgname" : "test_pbpolicy", "cn" : "pbtest", "role" : "primary" }'
```

Delete the Host Again

```
# pbdbutil --svc -d '{ "cn" : "pbtest" }'  
6024 Host is a primary server - please reassign before deleting the host
```

Delete the Service Group

```
# pbdbutil -svc -d '{ "svcgname" : "test_pbpolicy" }' --force
```


Synchronize Policy Configuration and Other Configuration Files

Although all configuration databases are automatically synchronized across the Service Group, other configuration such as Privilege Management for Unix and Linux policy scripts and encryption keys are not. They must be manually configured to synchronize across the Service Group. Only files that are kept within the standard configuration database **pb.db** on a primary server can be synchronized, so they need to be imported, and then synchronization configured.

All configuration databases are automatically synchronized across the Service Group. Other files, such as policy scripts and encryption keys, are not similarly treated and must be manually set up for synchronization. Only files that are kept within the standard configuration database **/etc/pb.db** on a primary server can be synchronized, so they need to be imported, and then synchronization configured.

The **pbdbutil** utility has been enhanced to provide the new synchronization options:

Usage

```
pbdbutil --cfg [<options>] [ <file> <file> ...]
```

-A <file> <svcname> <...>	Set file as being automatically synchronized within Service Group.
-X <file> <svcname> <...>	Unset file as being automatically synchronized within Service Group.
-L	List synchronization configuration for CFG files in the database.

Synchronize Privilege Management for Unix and Linux REST appkeys

Privilege Management for Unix and Linux REST appkeys are often required to authenticate users and services on remote servers, and are specific to each host. However, to provide role-based access to servers across a Service Group, REST appkeys can now be marked as synchronized across the Service Group.



Note: The host must be the primary of the specified Service Group to synchronize the appkeys.

Usage

```
pbdbutil --rest [<options>] [ <file> <file> ...]
```

-g <appid> [--svcname <name>] [<acl> ...]	Create new Application key with ACLs. Specify svcname to sync key across Service Group.
--	---

Database Synchronization

dbsyncdb

- **Version 9.3.0 and earlier:** **dbsyncdb** setting not available.
- **Version 9.4.0 and later:** **dbsyncdb** setting available.

The **dbsyncdb** option specifies the full path to the Database Synchronization Summary Database. This file is created in **datbasedir** by default, unless the file name starts with a slash (/).

**Example:**

```
dbsyncdb /etc/pbdbsync.db
```

Default

```
dbsyncdb /opt/<prefix>pbul<suffix>/dbs/dbsync.db
```

Used On

All primary servers when Registry Name Server is enabled.

dbsyncrefresh

- **Version 9.3.0 and earlier:** **dbsyncrefresh** setting not available.
- **Version 9.4.0 and later:** **dbsyncrefresh** setting available.

The **dbsyncrefresh** option defines the interval in seconds between database synchronization tasks. Increasing this value lowers the load on primary servers, but increases the time before configuration changes are applied to secondary servers.

**Example:**

```
dbsyncrefresh 360
```

Default

```
dbsyncrefresh 3600
```

Used On

All primary servers when Registry Name Server is enabled.

dbsyncloginterval

- **Version 9.3.0 and earlier:** **dbsyncloginterval** setting not available.
- **Version 9.4.0 and later:** **dbsyncloginterval** setting available.

The **dbsyncloginterval** option defines the interval in seconds between logging synchronization success and failure messages. Increasing this time makes the REST log smaller, but provides slower feedback on current status of the Database Synchronization on any given host.

**Example:**

```
dbsyncloginterval 360
```

Default

```
dbsyncloginterval 720
```

Used On

All primary servers when Registry Name Server is enabled.

Troubleshoot Registry Name Service and Database Synchronization Issues

The Registry Name Service and Database Synchronization are complex and there are many factors that can affect the smooth running of the service. Every host, including the Primary Registry Name Server, has a Registry Name Service Cache, which they use to look up all Service Group information that are applicable to them.

All of the servers that provide these services are implemented using the REST services (clients may or may not have REST services installed, but this should not be relevant to the overall service), and as such, one of the first checks is to make sure the relevant REST services are running. Use an appropriate **ps** (**man1**) command to show the process information:

```
# ps -ef | egrep "pblight|pbconfig"
root 124824 1 0 12:12 pts/1 00:00:00 /usr/lib/beyondtrust/pb/rest/sbin/pblighttpd-svc -d -i #mon
(wait)
root 124825 124824 0 12:12 pts/1 00:00:01 /usr/lib/beyondtrust/pb/rest/sbin/pblighttpd-svc -d -i
#sched (sleep)
pblight 124826 124824 0 12:12 pts/1 00:00:00 pblighttpd -D -m /usr/lib/beyondtrust/pb/rest/lib -f
/usr/lib/beyondtrust/pb/rest/etc/pblighttpd.conf
root 124827 124826 0 12:12 pts/1 00:00:00 /usr/lib/beyondtrust/pb/rest/sbin/pbconfigd
root 124828 124826 0 12:12 pts/1 00:00:00 /usr/lib/beyondtrust/pb/rest/sbin/pbconfigd
```

The number of **pbconfigd** processes running differs according to the host role within the enterprise.

If these services are not running, check the corresponding system logs and **pblighttpd**/REST logs for errors, and use the platform specific method of restarting the services.

Next, check that the host has the correct Service Group information.

Find out which Service Groups the host is a member of on the Primary Registry Name Server:

```
# pbdbutil -P --svc -L -L pbulprimrns
{
  "hostid": 1,
  "cn": "pbulprimrns",
  "uuid": "3d13a9eb-7340-4199-aa47-1570941bd50f",
  "fqdn": "pbulprimrns.org.com",
  "addrs": [
    {
      "family": 4,
      "addr": "192.168.1.1",
      "port": 24351
    }
  ],
  "tnlzone": 0,
  "updated_usec": "2016-11-11 12:03:47",
  "deleted": false,
  "svcs": [
    {
      "svcgid": 1,
      "hostid": 1,
      "role": "primary",
      "sorder": 1,
      "created_usec": "2016-11-11 12:03:47",
      "updated_usec": "2016-11-11 12:03:47",
    }
  ]
}
```

```
    "svcgname": "registry_name_service",
    "svc": "registry",
    "deleted": 0
  },
  {
    "svcgid": 4,
    "hostid": 1,
    "role": "secondary",
    "sorder": 2,
    "created_usec": "2016-11-11 12:04:35",
    "updated_usec": "2016-11-11 12:03:47",
    "svcgname": "dflt_sudopolicy_service",
    "svc": "sudopolicy",
    "deleted": 0
  }
]
```

Check the IP Addresses and List of Service Groups

Then check on the host to make sure the Registry Name Service Cache shows corresponding information:

```
# pbdbutil -P --scache -l
{
  "svcgname": "registry_name_service",
  "svc": "registry",
  "sorder": 1,
  "cn": "pbulprimrns",
  "uuid": "3d13a9eb-7340-4199-aa47-1570941bd50f",
  "fqdn": "pbulprimrns.org.com",
  "addrs": [
    {
      "family": 4,
      "addr": "192.168.1.1",
      "port": 24351
    }
  ],
  "role": "primary",
  "lastupdated_usec": "2016-11-11 12:03:47"
}
{
  "svcgname": "dflt_sudopolicy_service",
  "svc": "sudopolicy",
  "sorder": 1,
  "cn": "pbtest",
  "uuid": "12345676789",
  "fqdn": "pbtest",
  "addrs": [
    {
      "family": 4,
      "addr": "192.168.1.5",
      "port": 24351
    }
  ]
}
```

```

    ],
    "role": "primary",
    "lastupdated_usec": "2016-11-11 12:04:23"
  }
  {
    "svcgname": "dflt_sudopolicy_service",
    "svc": "sudopolicy",
    "sorder": 2,
    "cn": "pbulprimrns",
    "uuid": "3d13a9eb-7340-4199-aa47-1570941bd50f",
    "fqdn": "pbulprimrns.org.com",
    "addrs": [
      {
        "family": 4,
        "addr": "192.168.1.1",
        "port": 24351
      }
    ],
    "role": "secondary",
    "lastupdated_usec": "2016-11-11 12:04:35"
  }

```

If this information differs, and the **lastupdated** times are significantly different, refresh the cache:

```
# pbdbutil --scache -R
```

If this produces an error, then re-initialize the Registry Name Service Cache on the host using:

```
# pbdbutil --scache -N '{"hostname" : "<primaryRNS>, "appid": "<appid>": "appkey" : "<appkeys>"}'
--force
```

This clears the existing Registry Name Service Cache database and reloads it from the Primary Registry Name Server.

If databases are not correctly synchronizing across servers within the Service Group, firstly identify the primary server within that Service Group:

```
# pbdbutil -P --svc -g '{ "primary" : "registry_name_service" }'
{
  "svcgid": 1,
  "svcgname": "registry_name_service",
  "svc": "registry",
  "updated_usec": "2016-11-11 12:03:47",
  "deleted": false,
  "hostid": 1,
  "role": "primary",
  "sorder": 1,
  "created_usec": "2016-11-11 12:03:47",
  "cn": "pbulprimrns",
  "uuid": "3d13a9eb-7340-4199-aa47-1570941bd50f",
  "fqdn": "pbulprimrns.org.com",
  "addrs": [
    {
      "family": 4,
      "addr": "192.168.1.1",

```

```
    "port": 24351
  }
  ],
  "tnlzone": 0
}
```

Next, on that host list the contents of the Database Synchronization Summary Database:

```
pbdbutil --dbsync -l
{"cn":"pbtest","svc":"registry","dbid":1,"dbname":"/etc/pb.db","dbuuid":"9b332c26-b8bb-4546-9ed2-bf93146dd08c","lastupdated":"2016-11-11 15:10:44","lasttid":0}

{"cn":"pbtest","svc":"registry","dbid":2,"dbname":"/opt/pbul/dbs/pbrstkeys.db","dbuuid":"a22e8a75-fc6c-4f0e-ae9-b0d764c7e820","lastupdated":"2016-11-11 15:10:44","lasttid":0}
{"cn":"pbtest","svc":"registry","dbid":257,"dbname":"/opt/pbul/dbs/pbsvc.db","dbuuid":"97423b4f-2c5e-42c2-a87a-aeaaaa826c5b","lastupdated":"2016-11-11 15:12:14","lasttid":40}

{"cn":"pbtest","svc":"registry","dbid":258,"dbname":"/opt/pbul/dbs/pbregclnt.db","dbuuid":"f08b8679-8bcb-4f7d-b431-1bbab688e0c1","lastupdated":"2016-11-11 15:12:14","lasttid":0}
```

And then for each individual database check for outstanding transactions:

```
# pbdbutil --dbsync -l /opt/pbul/dbs/pbsvc.db
{"path":"/opt/pbul/dbs/pbsvc.db","sz":0}
```

If necessary, reset the Summary Database information and force a resynchronization:

```
# pbdbutil --dbsync -R registry
```

Finally, check that the Scheduling Service is running, and has recently run. This service provides the regular Registry Name Service Cache and Database Synchronization updates:

```
# pbdbutil --info --sched
{"id":"svccache_update","grp":"system","epoch":"2016-11-11 14:36:57","reoccurs":110,"retry":0,"backoff":0,"retried":0}
{"id":"database_sync","grp":"system","epoch":"2016-11-11 14:35:37","reoccurs":30,"retry":0,"backoff":0,"retried":0}
```

Logging

Privilege Management for Unix and Linux has a flexible logging capability that enables you to control what is logged, when, and where. There are four types of logs:


- Event logs
- I/O logs (sometimes called session logs)
- Diagnostic logs
- Debug trace logs

Event Logs

Event logs are produced by default and record the following information:

- Accept and Reject status
- The user's **pbrun** environment when an attempt is made to run **pbrun**
- Keystroke action events
- Task status (if task finished successfully or unsuccessfully)

Starting with v10.3.0, the event log can be a flat file, a local SQLite database, or sent to an external Oracle or MySQL database.

 For more information, please see "Auditing and Logging" on page 114.

Example of Event Log Entries



Example: The output from a **pblog** session that simply reads the event log is:

```
Accept 2018/09/20 11:37:57 chris@sparky -> chris@sparky
kill -9 570
Command finished with exit status 0
Accept 2018/09/20 11:38:09 chris@sparky -> chris@sparky
ls /export/home
Command finished with exit status 0
Reject 2018/09/20 11:37:39 chris@sparky
umount
```

Read Verbose Event Log Entries



Example: Given the following simple policy:

```
/* This is a test */
if (user == "sandy")
reject;
```

The output when **pblog -l** is run is as follows:

```
Reject 2010/06/23 11:33:34 sandy@octopus.company.com by root@octopus.company.com
ls
Request rejected by pbmasterd on octopus.company.com.

argc = 1
argv = {"ls"} bkgd = 0
clienthost = "octopus.company.com"
command = "ls"
cwd = "/var/log"
date = "2010/06/23"

day = 23 dayname = "Wed"
env = {"LANG=C", "PATH=./usr/local/bin", "EDITOR=vi", "LOGNAME=sandy",
"MAIL=/var/mail/sandy", "TERM=dtterm", "USER=sandy"}
event = "Reject"
eventlog = "/var/log/pb.eventlog"
exitstatus = "Request rejected by pbmasterd on octopus.company.com.[a]" false = 0
group = "uts" groups = {"uts"}
host = "octopus.company.com" hour = 11
i18n_date = "06/23/10" i18n_day = "23"
i18n_dayname = "Wed" i18n_hour = "11"
i18n_minute = "33"
i18n_month = "06"
i18n_time = "11:33:34"
i18n_year = "2010" iolog = ""
lineinfile = "/opt/pbul/policies/pb.conf"
linenum = "1"
localmode = 0
lognopassword = 1
logport = "32224" logservers = {"sandy"}
logstderr = 1
logstdin = 1
logstdout = 1
masterhost = "octopus.company.com"
masterlocale = "C"
minute = 33
month = 6
nice = 0 optarg = ""
opterr = 1
optimizedrunmode = 1
optind = 1
optopt = ""
```



```
optreset = 1
optstrictparameters = 1
pbclientmode = "run"
pbclientname = "pbrun"
pblogdmachine = "i686"
pblogdnodename = "sandy"
pblogdrelease = "2.6.22.5-31-default"
pblogdsysname = "Linux"
pblogdversion = "#1 SMP 2007/09/21 22:29:00 UTC"
pbmasterdmachine = "i686"
pbmasterdnodename = "sandy"
pbmasterdrelease = "2.6.22.5-31-default"
pbmasterdsysname = "Linux"
pbmasterdversion = "#1 SMP 2007/09/21 22:29:00 UTC"
pbrunmachine = "i686"
pbrunnodename = "sandy"
pbrunrelease = "2.6.22.5-31-default"
pbrunsysname = "Linux"
pbrunversion = "#1 SMP 2007/09/21 22:29:00 UTC"
pbversion = "6.1.0-15"
pid = 18511
psmcmapid = "7f0000024c22537e484D"
ptyflags = 7
rejectnullpasswords = 0 requestuser = "sandy" rlimit_as = -977616896
rlimit_core = 0
rlimit_cpu = -1
rlimit_data = -1
rlimit_fsize = -1
rlimit_locks = -1
rlimit_memlock = 32768
rlimit_nofile = 1024
rlimit_nproc = 15349
rlimit_rss = 1692928000
rlimit_stack = 8388608
runargv = {"ls"}
runbkgd = 0
runcommand = "ls"
runcwd = "/var/log"
runenablerlimits = 0
runenv = {"LANG=C", "PATH=./usr/local/bin", "EDITOR=vi", "LOGNAME=sandy",
"MAIL=/var/mail/sandy", "TERM=dtterm", "USER=sandy"}
rungroup = "sandy"
rungroups = {"sandy", "amanda"}
runhost = "octopus.company.com"
runlocalmode = 0
runnice = 0
runoptimizedrunmode = 1
runptyflags = 7
runrlimit_as = -977616896
runrlimit_core = 0
runrlimit_cpu = -1
runrlimit_data = -1
runrlimit_fsize = -1
```



```

runrlimit_locks = -1
runrlimit_memlock = 32768
runrlimit_nofile = 1024
runrlimit_nproc = 15349
runrlimit_rss = 1692928000
runrlimit_stack = 8388608
runsolarisproject = ""
runtimeout = 0
runtimeoutoverride = 0
runumask = 18 runuser = "sandy"
solarisproject = "" status = 0
submitthost = "octopus.company.com"
submitthostip = "127.0.0.2"
submitlocale = "en_US.UTF-8" submittpid = 18509
subprocuser = "sandy" time = "11:33:34"
timezone = "PDT" true = 1
ttyname = "/dev/pts/20"
umask = 18
uniqueid = "7f0000024c22537e484F"
user = "sandy"
year = 2010
    
```

*This log can be read as the request was rejected. The **reject** statement is on line 3 (**linenum**) in the file **/opt/pbul/policies/pb.conf** (**lineinfile**).*



Example: Given a policy where an Accept can happen:

```

/* Another test policy */
adminusers = {"sandy", "happy"};
okcommands = {"ls", "mount"};
if ((user in adminusers) && (command in okcommands))
{
    runuser = "root"; accept;
}
    
```

sandy executes the command:

```
pbrun ls
```

The event log from a **pblog -l** is similar to:

```

Accept 2010/06/23 11:43:44 sandy@octopus.company.com -> root@octopus.company.com by
octopus.company.com ls
Command finished with exit status 0 argv = 1
argv = {"ls"} bkgd = 0
clienthost = "octopus.company.com"
command = "ls"
cwd = "/tmp"
date = "2010/06/23" day = 23
dayname = "Wed"
    
```



```
env = {"LANG=C", "PATH=./usr/local/bin", "EDITOR=vi", "LOGNAME=sandy",
"MAIL=/var/mail/sandy", "TERM=dtterm", "USER=sandy"}
event = "Accept"
eventlog = "/var/log/pb.eventlog"
exitdate = "2010/06/23"
exitstatus = "Command finished with exit status 0"
exittime = "11:43:45"
false = 0 group = "uts"
groups = {"uts"}
host = "octopus.company.com"
hour = 11
i18n_date = "06/23/10" i18n_day = "23"
i18n_dayname = "Wed"
i18n_exitdate = "06/23/10"
i18n_exittime = "11:43:45"
i18n_hour = "11"
i18n_minute = "43"
i18n_month = "06"
i18n_time = "11:43:44"
i18n_year = "2010" iolog = ""
lineinfile = "/opt/pbul/policies/pb.conf"
linenum = "1"
localmode = 0
lognopassword = 1
logpid = 18829
logport = "32224" logservers = {"octopus"}
logstderr = 1
logstdin = 1
logstdout = 1
masterhost = "octopus.company.com"
masterlocale = "C"
minute = 43
month = 6
nice = 0
optarg = ""
opterr = 1
optimizedrunmode = 1
optind = 1
optopt = ""
optreset = 1
optstrictparameters = 1
pbclientmode = "run"
pbclientname = "pbrun"
pblogdmachine = "i686"
pblogdnodename = "octopus"
pblogdrelease = "2.6.22.5-31-default"
pblogdsysname = "Linux"
pblogdversion = "#1 SMP 2007/09/21 22:29:00 UTC"
pbmasterdmachine = "i686"
pbmasterdnodename = "octopus"
pbmasterdrelease = "2.6.22.5-31-default"
pbmasterdsysname = "Linux"
```



```
pbrunmasterdversion = "#1 SMP 2007/09/21 22:29:00 UTC"
pbrunmachine = "i686"
pbrunnodename = "octopus"
pbrunrelease = "2.6.22.5-31-default"
pbrunsysname = "Linux"
pbrunversion = "#1 SMP 2007/09/21 22:29:00 UTC"
pbversion = "6.1.0-15"
pid = 18824
psmcmapid = "7f0000024c2255e04986"

ptyflags = 7
rejectnullpasswords = 0 requestuser = "sandy"
rlimit_as = -977616896
rlimit_core = 0
rlimit_cpu = -1
rlimit_data = -1
rlimit_fsize = -1
rlimit_locks = -1
rlimit_memlock = 32768
rlimit_nofile = 1024
rlimit_nproc = 15349
rlimit_rss = 1692928000
rlimit_stack = 8388608
runargv = {"ls"} runbkgd = 0
runcommand = "ls"
runcwd = "/tmp"
runenablerlimits = 0
runenv = {"LANG=C", "PATH=./usr/local/bin", "EDITOR=vi", "LOGNAME=sandy",
"MAIL=/var/mail/sandy", "TERM=dtterm", "USER=sandy"}
rungroup = "uts"
rungroups = {"uts"}
runhost = "octopus.company.com"
runlocalmode = 0
runnice = 0
runoptimizedrunmode = 1
runpid = 18822
runptyflags = 7
runrlimit_as = -977616896
runrlimit_core = 0
runrlimit_cpu = -1
runrlimit_data = -1
runrlimit_fsize = -1
runrlimit_locks = -1
runrlimit_memlock = 32768
runrlimit_nofile = 1024
runrlimit_nproc = 15349
runrlimit_rss = 1692928000
runrlimit_stack = 8388608
runsolarisproject = ""
runtimeout = 0
runtimeoutoverride = 0
runumask = 18
```



```
runuser = "sandy"
solarisproject = ""
status = 0
submitthost = "octopus.company.com"
submitthostip = "127.0.0.2"
submitlocale = "en_US.UTF-8"
submitpid = 18822
subprocuser = "root"
time = "11:43:44"
timezone = "PDT" true = 1
ttyname = "/dev/pts/20"
umask = 18
uniqueid = "7f0000024c2255e04988"
user = "sandy"
year = 2010
```

The major differences between the output of this **pblog -v** example and the previous one are:

- The inclusion of the run variables (**runcommand**, **runuser**, **rungroup**, and so forth)
- The inclusion of the exit variables (**exittime**, **exitdate**, and an updated **exitstatus**)
- The user-defined variables **adminusers** and **okcommands**

By looking at the values in this output, you can determine the following:

- When a user ran a command (**time** and **date**)
- Where a user ran the command (**ttyname** and **submitthost**)
- The command that the user requested (**command**)
- Which command was actually run (**runcommand**)
- What user the command was run as (**runuser**)
- How it terminated (**exitstatus**)
- The value of locale settings (values of **LC_XXXX** environment variables) on the submitthost and Policy Server host (**submitlocale** and **masterlocale**)



Note: The value listed for these variables can differ depending on the platform, and also whether the **LC_XXXX** variables are all set to the same value.

For example, on a Linux platform when all **LC_XXXX** (**LC_CTYPE**, **LC_MONETARY**, **LC_TIME**, etc.) variables are set to the same value, **submitlocale** displays as "**C**"; however, on an HP or AIX platform, it displays as "**C C C C C**".

In addition on the same Linux platform where **LC_CTYPE** is set to "**POSIX**" for example and everything else is set to "**C**", then **submitlocale** displays as follows:

```
LC_CTYPE=POSIX;LC_NUMERIC=C;LC_TIME=C;LC_COLLATE=C;LC_MONETARY=C;LC_MESSAGES=C;LC_PAPER=C;LC_NAME=C;LC_ADDRESS=C;LC_TELEPHONE=C;LC_MEASUREMENT=C;LC_IDENTIFICATION=C
```

On HP with those same settings, **submitlocale** displays as "**C POSIX C C C C**".

Accept/Reject Logging

Event log entries are generated that note when a job is accepted or rejected and when an accepted job finishes running. The **pblog** program enables you to selectively choose and display log entries in a file that is specified by **eventlog**. The variable **event** is set to the event type (Accept, Reject, or Finish). **pblog**'s options enable you to specify only certain entries to print, set the output, and specify the format for each event type, or print all of the variables that are stored with each entry. When a job finishes running, the variable **exitstatus** describes how it finished; **exitdate** and **exittime** specify when it finished running.



Note: If the **exitstatus** is undefined, then the **exittime** is not logged to the event log. It thus becomes unavailable for the event on which the **exitstatus** is undefined.

Whenever an attempt is made to run a command through **pbrun**, the attempt is appended to the event log. The default location of the event log is usually **/var/adm/pb.eventlog**, **/usr/adm/pb.eventlog**, or **/var/log/pb.eventlog**, depending on your operating system. The location of this log file can be changed by setting the **eventlog** setting in the settings file or by using the **eventlog** variable in a policy file. The event log is a data file and any attempt to edit the file or it might corrupt it. To read the contents of an event log, execute the **pblog** command.

Keystroke Action Events

Privilege Management for Unix and Linux also records various keystroke action events in the event log. The administrator can specify that various words or patterns, when typed into a secured task, be logged to the event log and optionally terminate the job.

The easiest way to see the logged or rejected keystrokes in a session is to execute:

```
pblog
```

i To enable keystroke action events, please see the **setkeystrokeaction** procedure in the [Privilege Management for Unix and Linux Policy Language Guide](https://www.beyondtrust.com/docs/privilege-management/unix-linux/index.htm) at <https://www.beyondtrust.com/docs/privilege-management/unix-linux/index.htm>.

Reporting and Data Extraction

Privilege Management for Unix and Linux provides the ability to extract keyword data from the event log. This data can be displayed or used by other programs (database import programs, spreadsheets, and report writers, for example).

Use the **pblog -c** option to extract the required data and/or use various Unix/Linux search tools (**grep**, **awk**, **sed**) to extract and redirect the data to a file. Here is a shell script to format **pblog** output:

```
#!/usr/bin/sh
NOTE: only root can run pblog
echo " "
echo " "
echo " "
echo "Report: Successful Root-Level Events "
echo "----- "
echo " "
echo " "
echo "Submitted from: User Command"
pblog -c 'event=="Accept"' -a 'submithost + " " + user + " " + runcommand'
echo " "
echo " "
echo " "
echo "Report: Unsuccessful Attempts "
echo "----- "
echo " "
echo " "
pblog -c 'event=="Reject"'
```



Example: A sample report:

```
Report: Successful Root-Level Events
-----

Submitted from: User Command
sparky.company.com chris kill
sparky.company.com chris csh
sparky.company.com chris csh

Report: Unsuccessful Attempts
-----

Reject 1999/07/31 10:40:50 chris@sparky.company.com
kill -9 396
Reject 1999/07/31 10:44:10 chris@sparky.company.com
mount /dev/rmt01 /mnt
```

Merge Multiple Event Logs

If event logs are kept on different machines, then they can be merged into a single log by using the **pbsync -L** command. This feature is useful if there are separate dedicated log servers or if a secondary log server was used because of a log server failover.

Advanced Reporting

Privilege Management for Unix and Linux provides advanced event log reporting using the **pbreport** utility. After this utility is configured using the Privilege Management for Unix and Linux browser, it allows the system administrator to perform advanced reporting tasks such as extracting selected fields and designing customized reports.



For more information on this utility, please see the following:

- *"pbreport" on page 463*
- *[Privilege Management for Unix and Linux Browser Interface Guide](https://www.beyondtrust.com/docs/privilege-management/unix-linux/index.htm) at <https://www.beyondtrust.com/docs/privilege-management/unix-linux/index.htm>*

Event Log Rotation

Both **pbmasterd** and **pblogd** have the ability to rotate the event log. The current event log is renamed with a **date/time** tag, and a new event log is created with the specified event log name.

Event Log Rotation Based On Size

The **eventlogrotate** keyword specifies a rotate size and an optional path for the resulting rotated file. The size may include a **k**, **m**, or **g**, indicating the specified number is multiplied by **1024**, **1048576**, or **1073741824**, respectively. This mechanism rotates the event log only during Accept and Reject events. The optional path specifies a new path only (not the file name portion). The rotated event log has the file name format **pb.eventlog_YYYYMMDD_HHMMSS**.



Example:

```
eventlogrotate size=8k,path=/somewhere/else/
```

If the optional path is specified, the event log is renamed to the new file name (on the original file system) while the event log lock is held, then after the lock is released, the old renamed event log is moved to the optional path (possibly on a different file system). If the specified path does not exist, it is created.

This mechanism works with the event log **/path/filename** specified in the **pb.settings** file, or as changed via policy.

Variable Substitution in the Path

The path portion of the **eventlogrotate** keyword supports variable substitution. This is most useful to rotate to directories based on hostname or date. This mechanism works only when rotating by size during an Accept or Reject event. When rotating via command line (or cron), no policy variables are available. Variables used in the path portion of the **eventlogrotate** keyword must be variables that exist in every Accept and Reject event.



Note: The date variable results in a **yyyy/mm/dd** format, which includes subdirectories. For a **yyyymmdd** format without subdirectories, use **%year%%i18n_month%%i18n_day%**.



Example:

```
eventlogrotate "size=8k, path=/var/log/pb/eventlogs/%masterhost%/%date%/"
```

Manual Event Log Rotation

The **--rotate** option (**-R**), for both **pbmasterd** and **pblogd**, allows manual rotation, or rotation via cron, for the event log **/path/filename** specified in **pb.settings**. The **--rotate** option does not operate on event log **/path/filenames** changed via policy. The file name of a specific event log to rotate can be specified as an optional argument to the **--rotate** (**-R**) option. This mechanism uses the path element of the **eventlogrotate** keyword (ignoring any size specification).

**Example:**

```
pbmasterd --rotate
```

**Example:**

```
pbmasterd --rotate /var/log/pbeventlogA
```

**Example:**

```
pblogd --rotate
```

**Example:**

```
pblogd --rotate /var/log/pbeventlogA
```

Event Log Rotation Based On Age

Cron can be configured to execute either **pbmasterd** or **pblogd** with the **--rotate** option, to periodically rotate the event log.



For more information, please see the [Privilege Management for Unix and Linux Installation Guide](https://www.beyondtrust.com/docs/privilege-management/unix-linux/index.htm) at <https://www.beyondtrust.com/docs/privilege-management/unix-linux/index.htm>.

I/O Logs

Privilege Management for Unix and Linux provides the ability to log any input to, or output from, a secured job. This feature tracks everything that a user does in a Unix/Linux shell or terminal session, including the standard input, standard output, and standard error streams.



Note: Prior to v8.5, applications that use their own non-standard file descriptors, such as X-Windows applications, cannot be logged through the I/O logs. A feature was added in v8.5.0 to capture X11 applications. For more information, please see "[X Window System Session Capturing](#)" on page 568.

The contents of the I/O log file can also be limited to specific streams. The policy writer can limit the amount of data that is recorded from each of these streams.

I/O logging is configured in the policy file. Set the variable **iolog** to the name of a unique log file. If you want to turn off any of the standard streams, then you can set the variables **logstdin**, **logstdout**, and **logstderr** to **false**, as appropriate. After the command has started, you can use the **pbreplay** command to replay whatever was logged to that point in time, even if the session is still in progress.



Example: In this example, the policy file settings have been modified to ensure that all input and output is logged in **logtoday** when the **adduser** program is run through Privilege Management for Unix and Linux:

```
if(command=="adduser") {
  iolog=logmktemp("/var/log/pb.logs/adduser.XXXXXX");
}
```

We do not recommend the use of a hard-coded file name, because I/O logs are not overwritten. An attempt to write to an existing I/O log file results in an error.

The function **logmktemp()** is useful for generating unique file names for log files. Like the Unix/Linux **mktemp** library call, this function takes a template pattern as its argument and returns a string with the filled template, creating a unique file name.



Example: This example sets the file name for recording input and output to a uniquely named file in **/usr/adm**. The file name begins with **pboutput** and ends with a unique number that **logmktemp** substitutes for the X characters:

```
iolog=logmktemp("/usr/adm/pboutputXXXXXX");
```

Different machines might have different file system structures, so when using a log server, you should specify log files using the log server's file system. Environment variables are recorded in the I/O log file. If you want to suppress the logging of some of those environment variables, then use the **logomit** list. This variable is a list that contains the names of variables to omit from the log file. This feature is used when there is either no need to log certain variables or when it is wasteful to do so.

The **lognopassword** variable enables the security administrator to control the logging of non-echoed text (which is traditionally used for passwords) so that no one can see what is typed. If **lognopassword** is set to **true**, then only echoed text is logged. If **lognopassword** is set to **false**, then everything a user types is logged.

Some commands generate a large amount of output. Two examples are a tar of a complete file system or a daemon that sends output to **stdout**. It may be useful to limit the amount of output that is recorded in an I/O log for each command so that the logs do not get too large.

The number of characters that are logged from the **stdin**, **stdout**, and **stderr** streams can be limited.

**Example:**

```
logstdinlimit=150;
logstdoutlimit=200;
logstderrlimit=4096;
```

This example limits the I/O log to 150 characters for standard input, 200 characters for standard output, and 4096 characters for standard error.

If the command is interactive, such as with a shell, then the characters come from **stdin**, **stdout**, and possibly **stderr**. Each time the output stream changes, **stdout** to **stdin** or **stdout** to **stderr**, the amount that is logged returns to **0** (unlimited).

**IMPORTANT!**

To avoid file corruption, do not attempt to edit an I/O log file.

Password Logging



To control password logging, please see "passwordlogging" on page 140.

Display I/O Logs

The **pbreplay** command shows the contents of an I/O log. **pbreplay** has two modes:

- Interactive
- Batch

Interactive mode displays the **stdout** and **stderr** of any session. Batch mode can show the **stderr**, **stdout**, or **stdin** of a session. In batch mode it is also possible to specify a switch so that end-of-line characters are displayed as usual rather than just a carriage return, thereby preventing the output from overwriting itself.

The nature of I/O logging means that when a user has certain options enabled in some shells (**set -o vi** in the Korn shell, for example), the input keys are escape sequences. The shells interpret these and the output is displayed on **stdout** for a given input. Privilege Management for Unix and Linux does not interpret these input keystrokes. To determine what command is being entered, look at the **stdout** (**pbreplay -o**) and extract when a command is run.

I/O Logging and System Crashes

Should a log server crash or the network connection to the log server is lost, then Privilege Management for Unix and Linux attempts to fail over to an alternative log server. In this case, the start and end events for a task might be stored on two or more different machines. If you need to consolidate the event logs, then you can use **pbsyncd**.

Additionally, an I/O log could be split between two or more machines. I/O logs can be merged with **pbsync**.



For more information, please see "pbsyncd" on page 476.

Diagnostic Logging

Diagnostic messages can be logged in syslog and into Privilege Management for Unix and Linux diagnostic log files. The log files are specified by the settings keywords.



For more information, please see *"Diagnostic Logging"* on page 173.

Debug Trace Logging

If you encounter difficulties or problems with Privilege Management for Unix and Linux, you can enable certain components to produce trace logs associated with each secured task processing. A non-zero debug level from **1** (lowest) to **9** (highest) controls the depth and detail of the debug trace messages that is logged. A zero debug level means no tracing is performed. Currently, this feature is available in **pbrun**, **pbsh**, **pbksh**, **pbmasterd**, **pblocald**, and **pblogd**.

The resulting trace log file names have the format:

```
<prefix><PBUL_program_name><suffix>.<version>.debug.<PID>
```

and are created in the same location as the corresponding diagnostics log.

To ensure security, the debug trace logs are only generated if the parent directories specified in the relevant diagnostics log settings **pbrunlog**, **pbkshlog**, **pbshlog**, **pbmasterdlog**, **pblogdlog**, and **pblocaldlog** are owned and writable only by root.

When debug tracing the clients **pbrun**, **pbksh**, and **pbsh**, if the corresponding diagnostic keywords **pbrunlog**, **kshlog**, and **shlog** are not present in the settings file, Privilege Management for Unix and Linux attempts to use the directory location specified in **pblocaldlog** or **pbmasterdlog**.

When on-demand tracing is enabled from the clients (**pbrun**, **pbksh**, **pbsh**), the client and any associated daemon that participates in the secured task processing produce a trace log for that session.

The daemons, on the other hand, can be set up to always start in a trace-enabled state on a host using one of the following methods:

- Use the debug option of the server program for daemons that are either stand-alone or started by a superdaemon.
- Manually create a hidden file containing a numeric debug level (1-9) in specific locations:
 - **pbmasterd**: `/etc/.<prefix>pbmasterd<suffix>.debug.setting`
 - **pblogd**: `/etc/.<prefix>pblogd<suffix>.debug.setting`
 - **pblocald**: `/etc/.<prefix>pblocald<suffix>.debug.setting`
- If running **pbrun** as root, use options in **pbrun** to enable tracing for the appropriate daemon. To initiate and permanently turn on tracing for **pbmasterd**, **pblocald**, and **pblogd**, use **pbrun**'s **-d mlog**, **-d llog**, and **-d glog**, respectively. The appropriate `/etc/.*debug.setting` file is created.

To disable debug trace logging for the server programs, locate and delete the hidden debug setting files created from one of the steps mentioned above:

- **pbmasterd**: `/etc/.<prefix>pbmasterd<suffix>.debug.setting`
- **pblogd**: `/etc/.<prefix>pblogd<suffix>.debug.setting`
- **pblocald**: `/etc/.<prefix>pblocald<suffix>.debug.setting`



For more information, please see the following:

- ["Diagnostic Logging" on page 173](#)
- *For the syntax information of the specific Privilege Management for Unix and Linux program to learn how to turn on debug tracing:*
 - ["pbrun" on page 465](#)
 - ["Enable Debug Trace Logging for pbsh and pbksh" on page 363](#)
 - ["pbmasterd" on page 441](#)

i

- *"pblocald" on page 430*
- *"pblogd" on page 439*

Log Synchronization

Privilege Management for Unix and Linux supports redundant servers to allow for the continuity of the security rules in the event the policy server host becomes unavailable. In such an environment, the log file is created on the primary log server.

If the primary log server becomes unavailable, then transactions are no longer recorded on the primary log server, and the secondary server takes over to continue to log events and I/O streams.

When the network is restored, the primary policy server resumes logging to the destination system. Network administrators who want to monitor an I/O log or an event log would have to manually collect the logs from the primary and secondary servers, determine which occurred first, and finally merge and synchronize the logs.

The log synchronization feature reduces the workload for system administrators by providing automated log consolidation and merging of logs across the network using a single command.



For more information, please see the following:

- ["Log Synchronization" on page 169](#)
- ["pbsync" on page 473](#)
- ["pbsyncd" on page 476](#)

Log Cleanup and Rotation

The required retention period for log files varies from organization to organization. The log files are invaluable as security audit trails and system administration history logs.

The event log grows in length and contains more data than you see when executing a simple **pblog** command. Execute **pblog -l** and you will see a list of the variables. We suggest you run a nightly Unix/Linux **cron** (or similar) job to rename the event log.



Example:

```
mv pb.eventlog pb.eventlog.prev
```

or

```
mv pb.eventlog pb.eventlog(datestamp)
```

The I/O log files increase in number rather than length. Their number should also be controlled using Unix/Linux **cron**. For example, you might want to keep two weeks of I/O logs before removing them from the system.

Log Host File System Space

If left unchecked, log files consume an increasing amount of disk space on the log host. This degrades log host performance and results in the loss of log files if space runs out. The file system space on the log host can be controlled and the system can be configured to fail over to the next log server with the **logreservedfilesystems** and **logreservedblocks** settings.



For more information, please see "[logreservedfilesystems and logreservedblocks](#)" on page 142.

Log Archiving

Beginning with v9.0, Privilege Management for Unix and Linux provides a logfile tracking and archiving mechanism for I/O logs and event logs. Each log file created can have its location recorded in a centralized database for future searches. Log files can be archived from the original logserver hosts, for the purpose of freeing up space on the log servers or for consolidating logs on designated archive hosts.

The log archiving process is performed by hosts that are installed and configured with the server components. Those components mandatorily install the Privilege Management REST service which is essential in log file movement and tracking.

The hosts involved in log archiving are categorized as:

Log Server

The host where the event log or I/O log is created either by **pblogd** or **pbmaterd**.

The **pblogarchive** program found on this host initiates the log file movement. After a successful archive, the log file exists on the destination Log Archive Storage Server and is removed from this host.

The settings file on this machine requires **enablelogtrackingdb**, **logarchivehost**, and **logarchivedbhost**.

i For more information, please see the following:

- "**pblogarchive**" on page 483
- "**Settings**" on page 44

Log Archive Storage Server

The destination host of the archived logfiles.

The settings file on this machine requires **logarchivedir**, the main directory under which the archived logfiles are organized.

To support multi-tier archiving, this host can also function as a log server, transferring files to another archive server. In such case, the required settings mentioned above are needed as well.

i For more information, please see "**Settings**" on page 44.

Log Archive Database Server

The host where the centralized log tracking database is created and maintained.

The settings file on this machine requires **logarchivedb**, which is the path name of the SQLite database file.

We recommend you do not designate a primary log server/policy server as the database server to avoid degrading log host performance. Plan for growth of the database file, depending on the volume of log files that get created.

Regardless of function, all hosts involved in log archiving are required to have **pbrestport** configured in the settings file. Optionally, **pbresturi** may also be used.

i For more information, please see "**Settings**" on page 44.

Archive Encrypted Log Files

The log archiving feature supports encrypted logfiles.



IMPORTANT!

The log server sending files to be archived and the Log Archive Storage Server receiving the archived logs must use the same encryption algorithm and key.

For event logs, **eventlogencryption** setting must be the same on the source and the destination. For I/O logs, the **iologencryption** setting must be the same on the source and the destination. Otherwise, the log file transfer could fail.

Archive by Age

Archiving by age may be achieved by configuring **cron** to invoke the **pblogarchive** program.

Other Logging Limitations

Privilege Management for Unix and Linux has a local mode that can be used when an administrator needs only to monitor who started tasks, and whether or not they were approved. This feature reduces resource usage by making Privilege Management for Unix and Linux programs leave the job stream as soon as a secured task is approved. When using local mode, only the start of the secured task is logged.



For more information about local mode, please see the following:

- *"Optimized Run Mode Processing" on page 21*
- *"Local Mode Processing" on page 22*
- *"allowlocalmode" on page 106*

Integrate Elasticsearch and Logstash

As of PMUL 22.1, you can route events to Elasticsearch or Logstash instances located either on the customer premises or in the cloud.

You can use HTTP and HTTPS communication for both destination types. The HTTPS communication options for on-premises instances are different from their cloud counterparts.

The destination types support authenticated connections. Credentials are stored in the file `/opt/pbul/dbs/pbelkcred.db`.

Configure Elasticsearch and Logstash

There are three ways to configure Elasticsearch and Logstash instances:

- Elasticsearch over HTTP (on-premises instance)
- Elasticsearch over HTTPS (on-premises instance)
- Elasticsearch over HTTPS (cloud instance)

Elasticsearch over HTTP (On-Premises Instance)

To use this configuration, add `elkinstances` to `/etc/pb.settings` on the log server:

```
### Elasticsearch
elkinstances      elasticsearch=http://<elk-host-name>:9200
```

In this scenario, an Elasticsearch instance is configured on `elk-host-name` to listen on TCP:9200. When making a similar change in your setup, do the following:

1. Stop the `pblighttpd` service (`pblighttpd_svc.sh stop`).
2. Add your Elasticsearch server to `/etc/pb.settings`.
3. Start the `pblighttpd` service (`pblighttpd_svc.sh start`).
4. Run a `pbrun <command>` on any server to create and write events to your log server.

After performing the steps above, verify the Accept and Finish events are sent to Elasticsearch. To do this, query the Elasticsearch host using a command like the following:

```
curl -X GET "http://<elk-host-name>:9200/pmul-eventlog-*/_search?pretty" \
-H 'Content-Type: application/json' -d'
```



Example:

```
$ curl -X GET "http://example-host:9200/pmul-eventlog-*/_search?pretty" \
-H 'Content-Type: application/json' -d'
{
  "query": {
    "match_all": { }
  },
  "sort": [
```



```

    {
      "@timestamp": "desc"
    }
  ]
}
'

```

You will see a lot of output data, most of which is for the Accept event. If you don't see data, view the `/var/log/pbrest.log` file to ensure a problem is reported.


Elasticsearch over HTTPS (On-Premises Instance)

To configure an Elasticsearch on-premises instance over HTTPS:

- Create certificates on the Elasticsearch machine.
- Configure Elasticsearch to use the certificates.
- Configure PMUL to use HTTPS communication with the generated client certificates.
- (Optional) Configure Elasticsearch and PMUL to support authentication.

Create the Certificate File (certs.zip) on the Elasticsearch System

To create a certificate, you can use the following website for guidance: <https://www.elastic.co/blog/configuring-ssl-tls-and-https-to-secure-elasticsearch-kibana-beats-and-logstash#create-ssl>. Details are summarized in this section.



Note: Change `<elk-host-name>` to the name of your Elasticsearch server.

```

# cd /usr/share/elasticsearch
# service elasticsearch stop
# mkdir -p tmp/certs
# vim tmp/certs/instances.yml
# cat tmp/certs/instances.yml
instances:
  - name: 'elk-host-name'
    dns: [ 'elk-host-name' ]
  - name: 'client'
    dns: [ 'client' ]
# ./bin/elasticsearch-certutil cert --keep-ca-key --pem --in tmp/certs/instances.yml \
--out tmp/certs/certs.zip
# cd tmp/certs
# unzip certs.zip
# mkdir /etc/elasticsearch/certs
# cp ca/ca.crt /etc/elasticsearch/certs
# cp <elk-host-name>/<elk-host-name>.crt /etc/elasticsearch/certs
# cp <elk-host-name>/<elk-host-name>.key /etc/elasticsearch/certs

```

Configure Elasticsearch to Use Certificates

1. Based on the above commands, add the following lines to `/etc/elasticsearch/elasticsearch.yml` to use HTTPS:

```
#
# ----- Security -----
#
xpack.security.http.ssl.enabled: true
xpack.security.http.ssl.key: certs/elk-host-name.key
xpack.security.http.ssl.certificate: certs/elk-host-name.crt
xpack.security.http.ssl.certificate_authorities: certs/ca.crt
```

2. Run **service elasticsearch start** as the root user. If that fails, run **service elasticsearch status** and follow the suggested steps.
3. When Elasticsearch starts correctly, verify insecure HTTPS connectivity from the log server using a command like the following (and seeing the displayed response):

```
$ curl --insecure -X GET https://<elk-host-name>:9200/?pretty \
-H 'Content-Type: application/json'
{
  "name" : "elk-host-name",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "u1EW_94MRCGnQkvo02tYdQ",
  "version" : {
    "number" : "7.14.1",
    ...
  },
  "tagline" : "You Know, for Search"
}
```

4. After **--insecure** works, use the certificate authority (CA) certificate. Copy the CA certificate from the Elasticsearch server to the PMUL log server using commands similar to the following:

```
# mkdir /opt/pbul/certs
# cd /opt/pbul/certs
# scp root@<elk-host-name>:/usr/share/elasticsearch/tmp/certs/ca/ca.crt
```

5. Use curl with the CA certificate:

```
$ curl -X GET https://<elk-host-name>:9200/?pretty \
--cacert /opt/pbul/certs/ca.crt -H 'Content-Type: application/json'
```



Note: Running **--cacert** is the same output as running **--insecure**.

Deploy the Client Certificates to PMUL

1. With insecure communication established, and the CA certificate shown to work, migrate the created client certificates to PMUL, and then test with curl. The curl command returns the same data as shown in the previous procedure.

```
# scp root@<elk-host-name>:/usr/share/elasticsearch/tmp/certs/client/client* .
# curl -X GET https://<elk-host-name>:9200/?pretty \
  --cacert /opt/pbul/certs/ca.crt --cert /opt/pbul/certs/client.crt \
  --key /opt/pbul/certs/client.key -H 'Content-Type: application/json'
```

2. Modify the Elasticsearch section of `/etc/pb.settings` as shown:

```
### Elasticsearch
elkinstances          elasticsearch=https://<elk-host-name>:9200
elkcafile             /opt/pbul/certs/ca.crt
elkcertfile          /opt/pbul/certs/client.crt
elkkeyfile           /opt/pbul/certs/client.key
```



IMPORTANT!

You must ensure that the host name in the `elkinstances` URL exactly matches the name in the `instances.yml` file from which the certificates were created. Communication will not work if you mix-and-match FQDN and short names. For example, `<elk-host-name>.pmul.net` in `pb.settings` and `<elk-host-name>` in `instances.yml`.

3. Restart `pblighttpd` (`pblighttpd_svc.sh restart`) and run `pbrun`. To verify that the Accept and Finish events are delivered, view the last few lines of `/var/log/pbrest.log` to ensure delivery succeeded and to get the last few characters of the event's unique ID.
4. Run the following command to ensure the event was delivered to the log server:

```
$ curl -X GET "https://<elk-host-name>:9200/pmul-eventlog-*/_search?pretty" \
  --cacert /opt/pbul/certs/ca.crt --cert /opt/pbul/certs/client.crt \
  --key /opt/pbul/certs/client.key -H 'Content-Type: application/json' -d'
{
  "query": { "match_all": { } },
  "sort": [ { "@timestamp": "desc" } ]
}
' | grep <unique_id>-
```

The output displays two lines with the IDs of the Accept and Finish events. To display more information, eliminate the `grep`.

Enable Authentication (Optional)

This step can be done before deploying client certificates. Assuming certificates are in place, temporarily disable certificates and HTTPS and configure authentication over HTTP.

1. On the Elasticsearch machine, stop Elasticsearch: **service elasticsearch stop**.
2. Modify the **Security** settings in `/etc/elasticsearch/elasticsearch.yml`:

```
#
# ----- Security -----
#
xpack.security.enabled: true
# xpack.security.http.ssl.enabled: true
```

```
# xpack.security.http.ssl.key: certs/<elk-host-name>.key
# xpack.security.http.ssl.certificate: certs/<elk-host-name>.crt
# xpack.security.http.ssl.certificate_authorities: certs/ca.crt
```

- Restart Elasticsearch: **service elasticsearch start**, and then create some authenticated users and passwords:

```
# cd /usr/share/elasticsearch
# bin/elasticsearch-setup-passwords interactive
```

- You are prompted to enter passwords for several built-in Elasticsearch users (including the elastic user). Make a note of the passwords.
- Use curl to test authentication from the log server. In the case where the password for the elastic user is **elastic**, you can use the following command:

```
$ curl -X GET "http://<elk-host-name>:9200/pmul-eventlog-*/_search?pretty" \
-u elastic:elastic -H 'Content-Type: application/json' -d'
{
  "query": { "match_all": { } },
  "sort": [ { "@timestamp": "desc" } ]
}
```

- Any messages from Elasticsearch previously stored in the database are displayed.
- Configure PMUL to authenticate (initially via HTTP) to deliver events to Elasticsearch. Add an Elasticsearch credential to PMUL:

```
# pbrestcall -l -X PUT -a <appid> -k <key> \
https://localhost:24351/REST/elkcred/elastic_basic -d \
'{"id": "elastic_basic", "type": "basic", "username": "elastic", \
"password": "elastic"}'
```

- The command replies with a JSON message that includes { **"status": 0** }.
- Set up PMUL to use the credential. Modify the **/etc/pb.settings** file on the log server as shown:

```
### Elasticsearch
elkinstances          elasticsearch=http://<elk-host-name>:9200
elkcredential         elastic_basic
# elkcafile           /opt/pbul/certs/ca.crt
# elkcrtfile         /opt/pbul/certs/client.crt
# elkkeyfile         /opt/pbul/certs/client.key
```

- Restart the pblighttpd service (**pblighttpd_svc.sh restart**) and execute a pbrun command that sends an event to the log server.
- Inspect **/var/log/pbrest.log** to ensure that no error occurred. If no error occurred, then query the Elasticsearch server to ensure the event was delivered there. Obtain the last few characters of the event's unique ID and issue a command such as the following:

```
$ curl -X GET "http://<elk-host-name>:9200/pmul-eventlog-*/_search?pretty" \
-u elastic:elastic -H 'Content-Type: application/json' -d'
{
```

```
"query": { "match_all": { } },
"sort": [ { "@timestamp": "desc" } ]
}
' | grep <uniqueid>-
```

12. Add HTTPS and certificates back into the configuration. To start, go to the Elasticsearch server and modify the **Security** settings in **/etc/elasticsearch/elasticsearch.yml** to re-enable secure communications:

```
#
# ----- Security -----
#
xpack.security.enabled: true
xpack.security.http.ssl.enabled: true
xpack.security.http.ssl.key: certs/<elk-host-name>.key
xpack.security.http.ssl.certificate: certs/<elk-host-name>.cert
xpack.security.http.ssl.certificate_authorities: certs/ca.crt
```

13. Restart Elasticsearch (**service elasticsearch restart**) and check the communications configuration from the log server side:

```
$ curl -X GET "https://<elk-host-name>:9200/pmml-eventlog-*/_search?pretty" \
--cacert /opt/pbul/certs/ca.crt --cert /opt/pbul/certs/client.crt \
--key /opt/pbul/certs/client.key -u elastic:elastic \
-H 'Content-Type: application/json' -d'
{
  "query": { "match_all": { } },
  "sort": [ { "@timestamp": "desc" } ]
}
' | grep <unique_id>-
```

14. Reconfigure PMUL on the log server to use certificates with authentication. Modify **/etc/pb.settings** with the Elasticsearch configuration as shown:

```
### Elasticsearch
elkinstances          elasticsearch=https://<elk-host-name>:9200
elkcredential         elastic_basic
elkcafile             /opt/pbul/certs/ca.crt
elkcertfile          /opt/pbul/certs/client.crt
elkkeyfile            /opt/pbul/certs/client.key
```

15. Restart pblighttpd (**pblighttpd_svc.sh restart**) on the log server and issue a pbrun command that generates a log event. Obtain that event ID from **/var/log/pbrest.log** and ensure that it was delivered to Elasticsearch. You can do this with the curl command shown above but grepping for a different unique_id. You will see two lines of output: one for the Accept event and the other for the Finish event.

Add Token and API Key Credentials

Elasticsearch supports authentication with tokens and API keys.

- Token authentication: The user submits a username and password, and Elasticsearch replies with a token that is used in subsequent requests without the username and password

- API key requests: The Elasticsearch administrator supplies the API ID and key.

Adding a token credential is like adding a basic credential, but requires an additional endpoint argument as shown:

```
# pbrestcall -l -X PUT -a <appid> -k <key> \
https://localhost:24351/REST/elkcred/elastic_token -d \
'{"id": "elastic_token", "type": "token", "username": "elastic", \
"password": "elastic", "endpoint": "_security/oath2/token"}'
```

Adding an API key (type **apikey**) credential is like adding a basic credential, but the arguments are different:

```
# pbrestcall -l -X PUT -a <appid> -k <key> \
https://localhost:24351/REST/elkcred/elastic_apikey -d \
'{"id": "elastic_apikey", "type": "apikey", "apiid": "<apiid>", \
"apikey": "<apikey>"}'
```

The values of **<apiid>** and **<apikey>** are provided by the Elasticsearch system administrator. It is possible to obtain the keys for testing using curl:

```
# curl -u <username>:<password> -X POST <elastic_url>/_security/api_key?pretty \
-H 'Content-Type: application/json' -d '{"name": "elastic_apikey" }'
```

```
{
  "id" : "bTltoX0BTJT5yB8Hshto",
  "name" : "elastic_apikey",
  "api_key" : "6Y2qXmHgSsKaQN-pB6D5TQ"
}
```

Cloud Instance (Elasticsearch over HTTPS)

You need the following information from the cloud provider or someone in your organization:

- URL (for example, <https://d6a3bf3dezr92180de2182410be.us-east-1.aws.found.io>)
- Username
- Password

Elasticsearch Keywords

elkinstances

- **Version 22.1 and later:** **elkinstances** setting available.

Events are sent to the URLs in elkinstances until an attempt succeeds. No subsequent URLs are attempted following a successful send.

Syntax

```
elkinstances [elasticsearch|logstash]=url1,...,urlN
```


where the URL items are **http** or **https** URLs that specify the endpoint up to and including the port number.

**Example:**

```
elkinstances    elasticsearch=https://elastic.io,http://localhost:9200
```

elkcredential

- **Version 22.1 and later:** **elkcredential** setting available.

Set the identifier of a stored credential to authenticate to the endpoints in `elkinstances`. The credential, if specified, is applied to all the endpoints.

Syntax

```
elkcredential <credential_id>
```

elkcafile

- **Version 22.1 and later:** **elkcafile** setting available.

The name of a copy of the certificate authority, in PEM format, file used by the remote Elasticsearch or Logstash server. This is used only for communicating with on-premises instances.

Syntax

```
elkcafile    <filename>
```

elkcertfile

- **Version 22.1 and later:** **elkcertfile** setting available.

A client certificate, in PEM format, counterpart of the server certificate used by Elasticsearch or Logstash. This is used only when communicating with on-premises instances.

Syntax

```
elkcertfile  <filename>
```

elkkeyfile

- **Version 22.1 and later:** **elkkeyfile** setting available.

The client key file, in PEM format, associated with **elkcertfile**. This differs from the private key present on the Elasticsearch or Logstash server. This is used only when communicating with on-premises instances.

Syntax

```
elkkeyfile <filename>
```

elkdatatypes

- **Version 22.2 and later:** **elkdatatypes** setting available.

Set one or more types of data to deliver to Elasticsearch. Current options are eventlog (Accept, Reject, Finish, Keystroke) and iolog.

The default is to send only eventlog data to Elasticsearch if PMUL is configured to communicate with Elasticsearch (that is, **elkinstances** and **elkcredential** are configured).

Syntax

```
elkdatatypes eventlog iolog
```

elkiologfieldsizekb

- **Version 22.2 and later:** **elkiologfieldsizekb** setting available.

The size, in kilobytes, of the amount of iolog session data that pbreplay writes to each chunk, in cases where pbreplay sends data to Elasticsearch in multiple chunks. The default is 1024, in which case pbreplay sends iolog data to Elasticsearch in 1MB (1024 * 1024) chunks.

Syntax

```
elkiologfieldsizekb N
```

Acceptable values for **N** range from 8 (for 8 KB) to 65536 (for 64 MB).

Default Value

1024

elasticsearchidxtemplate

- **Version 22.2 and later:** **elasticsearchidxtemplate** setting available.

The path to a template file that can be used to specify both default handling and handling of specific variables that PMUL sends to Elasticsearch.

Syntax

```
elasticsearchidxtemplate [absolute or relative file path]
```

Default Value

`/opt/pbul/elk/etc/pbelasticsearchtemplate.json`

elkecsconfiguration

- **Version 22.2 and later:** `elkecsconfiguration` setting available.

The main role of this file (the default is `/opt/pbul/elk/pbelkecsconfiguration.json`) is to set the mapping of PMUL field names to Elasticsearch Elastic Common Schema (ECS) field names. This is done in the `mappedFields` JSON object in the file. The objects in the file are described here:

- **"version": <number>**

The internal version number for the JSON file. The current value is 1. It might be used in later versions of PMUL.

- **"mappedFields": { ... }**

Contains entries that map PMUL fields to one or more ECS fields. For example, the entry:

- **"user": ["related.user", "user.name"]**

maps the PMUL variable `user` to ECS variables `related.user` and `user.name` – both ECS fields are populated with the value of the PMUL `user` variable. An ECS variable typically receives the value of the last PMUL variable written to it during processing. There are, however, three special ECS variables – `related.user`, `related.ip`, and `related.hosts` – that store all PMUL values sent to them in arrays. The `related.hosts` variable, for example, might contain the values for PMUL variables `logserver`, `masterhost`, `runhost` and `submithost`.

- **"subsetFields": { ... }**

Most PMUL fields are not written to Elasticsearch for **iolog Finish** events that have a corresponding **Accept** event. This is because most such fields duplicate their values in the **Accept** event. However, you might want a subset of some of the fields to be included in the **iolog** event sent to Elasticsearch anyway. An entry such as:

- **"logserver": true**

ensures the variable `logserver` is sent to Elasticsearch for both **Accept** and **Finish** events.

- **"excludeFields": { ... }**

Use to exclude unwanted fields from the Elasticsearch record. The fields are not populated. You can, however, exclude the PMUL fields `true` and `false` with the entries:

- **"true": true,**
- **"false": true**

in which case they are not forwarded to Elasticsearch.

Default Configuration File

/opt/pbul/elk/etc/pbelkecsconfiguration.json

elkdeliverytimeout

- **Version 22.1 and later:** `elkdeliverytimeout` setting available.

The number of seconds within which the event must be delivered to Elasticsearch or Logstash. The default is 30. The minimum is 0 (no timeout) and the maximum is 120.

Syntax

```
elkdeliverytimeout <value>
```

elkindexppattern

- **Version 22.1 and later:** `elkindexppattern` setting available.

Use an index pattern to set default index suffixes for event and iolog messages delivered to Elasticsearch.

- `%fieldN%` is used to define substitutions: `%year%`, `%month%`, and `%day%` are placeholders for today's year, month and day.
- `%<pmul-field-name>%` is a substitute for a PMUL field value, e.g., `%user%` for the value of the field user.

In cases where a non-existent or missing variable is used, the index defaults to one relevant to the current date (e.g., `pmul-eventlog-ecs-20220531`). Similar action is taken in cases when the syntax is used incorrectly.

We do not recommend using a pattern for eventlog; a field available in the Accept event (for example, user) might not exist in the Finish event. In this case, Elasticsearch cannot process Accept and Finish events to a single object.

Syntax

```
elkindexppattern iolog=%field1%..%fieldX% eventlog=%field1%..%fieldY%
```

elkoptions

- **Version 22.1 and later:** `elkoptions` setting available.

Configure additional options.

Valid Values

- **batchsize:** Sets the batch size for reading rows from the SQLite cache database.


Example:

```
elkoptions batchsize=20
```

siemcachedb

- **Version 22.1 and later:** **siemcachedb** setting available.

The **siemcachedb** setting specifies the absolute path of the SQLite database created by message router to store the events that did not make it to Elasticsearch or Logstash. The database is rotated to the **<dequeuedatasedir>/mrsiem** directory on reaching the limit set by the setting **siemcachedblimit**.

Default Value

```
/opt/pbul/dbs/pbsiemcache.db
```

Where **/opt/pbul/dbs** is the value from the setting **datasedir**.

siemcachedblimit

- **Version 22.1 and later:** **siemcachedblimit** setting available.

The **siemcachedblimit** setting sets the maximum database size of **siemcachedb** and also the interval in which the **siemcachedb** is rotated to **<dequeuedatasedir>/mrsiem** directory.

Syntax

```
siemcachedblimit size=<size-n>[K|M|G] limit=<time-x>[m|h]
```

Where:


- **size-n** is the maximum size limit of the **siemcachedb** database in KB [K] or MB [M] or GB [G].
- **time-x** is the interval in minutes [m] or hour [h] on which the **siemcachedb** database is rotated.

Examples



Example: When **size=0K**, there is no size limit set on the database but database is rotated for every 10 minutes:

```
siemcachedblimit size=0K limit=10m
```

 **Example:** In this example, the database is rotated on reaching the size limit of 1GB or the time interval of 2 hours, whichever is the first:

```
siemcachedblimit    size=1G limit=2h
```

Default Value

```
siemcachedblimit    size=0K limit=10m
```

siemdbencryption

- **Version 22.1 and later:** **siemdbencryption** setting available.

The **siemdbencryption** setting specifies the encryption for **siemcachedb** and the dequeue databases created by Privilege Management for Unix and Linux.

Syntax

```
siemdbencryption <algorithm-1>:<keyfile=/fullpath/data-file-1>  
[:<startdate=yyyy/mm/dd>:<enddate=yyyy/mm/dd>] <algorithm-2>:<keyfile=/fullpath/data-file-2>  
[:<startdate=yyyy/mm/dd>:<enddate=yyyy/mm/dd>] ...
```

Where:

- **algorithm-n** is the name of the algorithm type.
- **/fullpath/data-file** (optional) specifies the full path and file name of the data file, which is used to dynamically derive the encryption key.
- **startdate=yyyy/mm/dd** specifies the earliest date that this algorithm is to be used.
- **enddate=yyyy/mm/dd** specifies the latest date that this algorithm is to be used.

Default Value

None

dequeuedatabasedir

- **Version 22.1 and later:** **dequeuedatabasedir** setting available.

The **dequeuedatabasedir** setting is the absolute path of the directory in which subdirectory **mrsiem** is created by message router. The **siemcachedb** is rotated to this directory. A scheduled pbconfigd process attempts to dequeue the databases in **dequeuedatabasedir/mrsiem** for every **siemdqrefreshtime** interval.

Default value of the setting **dequeuedatabasedir** is dependant on the setting **basedir**.

Default Value

```
dequeuedatabasedir /opt/pbul/dequeuedbs
```

siemdqrefreshtime

- **Version 22.1 and later:** `siemdqrefreshtime` setting available.

The `siemdqrefreshtime` setting specifies the interval in minutes between the scheduled pbconfigd database dequeue tasks.

Default Value

```
siemdqrefreshtime 10
```

Credential Store

Starting in PMUL 22.1, a credential store is available. The credential store is an encrypted SQLite database stored in the file `/opt/pbul/dbs/pbelkcred.db`. Initially, the file has only one table with the following schema:

```
CREATE TABLE elkcreds (id TEXT PRIMARY KEY, value TEXT UNIQUE NOT NULL)
```

The `id` field is a simple string referenced by the text that optionally follows a URL in the `elkinstances` setting described previously. The `value` field contains a JSON fragment that describes one of the following credential types:

- Basic
- Token
- API key

Basic Authentication

This is a simple combination of username and password, with the format:

```
{ "id": "<id>", "type": "basic", "username": "<username>", "password": "<password>" }
```

It can be used to authenticate to both cloud and on-premises instances, but most likely will only be used for cloud instances. The username and password are included (though typically within HTTPS) with every event submitted to Elasticsearch or Logstash.

Elasticsearch and Logstash support basic authentication.

Token Authentication

For token authentication, PMUL would not send the username and password with every request. Rather, it would send the username and password only to obtain a token that can be used for subsequent requests until the token expires. The expiration would be given in the reply to the query that includes the username and password.

The format of the `value` field for token authentication:

```
{ "id": "<id>", "type": "token", "username": "<username>",  
  "password": "<password>", "endpoint": "<url_or_path>" }
```

The endpoint can either be a complete URL (starts with https://) or a path to be appended to the URL specified in the **elkinstances** field described above (for example, `/_security/oauth2/token`).

Token authentication is typically used with cloud instances but can be used with on-premises instances configured for token support.

Logstash does not support token authentication.

API Key Authentication

API key authentication is similar to token authentication but provides the ability to request the time to expire.

The API key authentication format:

```
{ "id": "<id>", "type": "apikey", "username": "<username>", "password": "<password>",
  "endpoint": "<url_or_path>", "expiration": "<valid_expiration>" }
```

The **valid_expiration** field can be a value like **1d** to indicate that the returned API key is valid for one day. The field is required, otherwise created API keys are permanent.

Logstash does not support API key authentication.

Integrate I/O Logging and Elasticsearch

Starting in PMUL 22.2, the `pbreplay` program can be configured to send messages to Elasticsearch and Logstash as well as Solr.

The `pbreplay` program cannot send events to both Elasticsearch and Solr. If a log server has a valid configuration for sending iologs -- `/etc/pb.settings` contains a valid value for the **elkinstances** setting and includes `iolog` in its **elkdatatypes** setting – then iologs are sent to Elasticsearch only, regardless of whether Solr is configured or configured correctly.

Requirements

The destination Elasticsearch system must be version 7.10 or later.

PMUL to Elasticsearch Field Mappings

The file `pbelkecsconfiguration.json` contains standard mappings of PMUL fields to Elastic Common Schema (ECS) fields.

The iolog session text is not mapped to an ECS field. These sessions can contain a lot of data, and therefore must be of the wildcard type. Since the session data can be quite large, a single iolog session may result in many individual messages sent to Elasticsearch. In such cases, the unique ID (mapped to the ECS field **event.id**) will have a chunk number added in the format **\$\$\$\$\$<chunk-number>\$\$\$**.

Mapping of PMUL Fields to ECS Fields

Fields are mapped in the **mappedFields** setting in the `elkscsconfiguration` file.

The PMUL field on the left maps to one or more ECS fields, as indicated by arrays, on the right.

```
"mappedFields": {
  "browserhost": ["related.hosts"],
  "browserip": ["related.ip"],
  "clienthost": ["related.hosts"],
  "event": ["event.action"],
```



```

"eventlog":          ["log.file.path"],
"group":            ["group.name"],
"host":            ["related.hosts"],
"logdversion":     ["agent.version"],
"loghostip":       ["related.ip"],
"loghostname":     ["agent.name", "log.logger", "related.hosts"],
"logserver":       ["agent.name", "log.logger", "related.hosts"],
"masterhost":     ["related.hosts", "server.domain"],
"masterhostip":   ["related.ip", "server.address", "server.ip"],
"pbclientkerberosuser": ["related.user"],
"pblogdrelease":  ["os.kernel"],
"pblogdsysname":  ["os.name"],
"pbrisklevel":    ["event.risk_score"],
"requestuser":    ["related.user"],
"runargv":        ["process.args"],
"runcommand":     ["process.name"],
"runconfirmuser": ["related.user"],
"runeffectiveuser": ["related.user"],
"runhost":        ["host.hostname", "host.name", "destination.domain",
"related.hosts"],
"runhostip":      ["host.ip", "destination.ip", "destination.address",
"related.ip"],
"runpid":         ["process.pid"],
"runuser":        ["related.user", "user.effective.name"],
"runutmpuser":    ["related.user"],
"submithost":     ["client.domain", "related.hosts", "source.domain"],
"submithostip":  ["related.ip", "client.address", "client.ip",
"source.ip", "source.address"],
"subprocuser":    ["related.user"],
"timezone":       ["event.timezone"],
"user":          ["related.user", "user.name"]
},

```

There are some PMUL fields (particularly in Accept events) with no direct mapping to an ECS field. The fields will be mapped to a non-ECS field **beyondtrust_com.pmul.<fieldname>**. For example, **beyondtrust_com.pmul.optimizedrunmode** for the PMUL **optimizedrunmode** field.

PMUL fields that are mapped to ECS fields are also mapped to a non-ECS **beyondtrust_com.pmul...** field.

Hard-coded Mappings to ECS Fields

PMUL Entity	ECS Field Name
PMUL event time (submit_utc, logaccept_utc, logreject_utc, logfinish_utc, logkeystroke_utc, runfinish_utc)	@timestamp
hardcoded string "logserver"	agent.type
ECS Schema Version (hardcoded)	ecs.version
iolog file name	log.file.path
Number of command line arguments	process.args_count
PMUL event finish time (runfinish_utc, logfinish_utc or last timestamp in iolog)	process.end
PMUL event start time (submit_utc, keystroke_utc, logaccept_utc, logreject_utc)	process.start

Migrate Existing PMUL Field Values to ECS Format

Starting with version 22.1, event data can be sent to Elasticsearch and Logstash. That data, however, was not delivered in the Elastic Common Schema (ECS) format.

Starting with version 22.2, event logs (and optionally iologs) are delivered in ECS format.

You can use the **btelkecsconvert** migration tool to convert existing non-ECS documents added in version 22.1 to ECS format supported in version 22.0 (and later).

Install and Run the Migration Tool



Note: The migration tool is only available through BeyondTrust Technical Support.

The migration tool file is a self-contained executable with no external dependencies. The migration tool is available for either Windows or Linux x86_64 operating systems.

Run the command against the Elasticsearch instance, as shown here:

```
C:\...\btelkecsconvert> btelkecsconvert -d yes -p pmul -u **:** http://host-name.pmul.net:9200
INFO: 2022/06/13 10:20:06 btelkecsconvert.go:370: found 1 index(es) with non-ECS document(s)
INFO: 2022/06/13 10:20:06 btelkecsconvert.go:388: Migrating 1 non-ECS document(s) from index
pmul-eventlog-20220613 to index pmul-eventlog-ecs-20220613
INFO: 2022/06/13 10:20:07 btelkecsconvert.go:668: reply: created: 1, updated: 0
INFO: 2022/06/13 10:20:09 btelkecsconvert.go:682: deleted index pmul-eventlog-20220613 and its
contained document(s)
C:\...\btelkecsconvert>
```

Arguments

-d, --delete-old-indexes	Required. Valid options are yes and no . <ul style="list-style-type: none"> Enter yes to delete old indexes and documents for which all documents within the index were successfully converted to ECS format. Enter no to retain the old indexes and documents.
-p, --product	Required. Enter the name as pmul .
-u, --user <credentials in username:password format>	Optional. Enter login credentials for the Elasticsearch instance.
URL	Required. The HTTP URL of the Elasticsearch instance. (e.g., http://elkhost:9200).
-v, --version	Optional. Prints the version.
-h, --help	Optional. Displays the Usage message to the console.

Message Router Troubleshooting

The Message Router provides a method of consuming and storing large amounts of different types of messages from various Privilege Management services. For example, **pbmasterd** and **pblogd** services employ it to write event data to the event log and to other integrated products such as BeyondInsight and Solr. This increases the performance when systems are loaded, while reducing load on the system. However, this also increases the complexity of the overall system. This means that while the service is running smoothly, no maintenance is required. However, if there is a problem with the service, other vital services may be affected.

The **pbadmin** command is used to monitor normal statistics for the Message Router.



Example: This example output is typical, and shows the Message Router is running and is receiving requests.

```
pbadmin -P --info --msgs
{
  "pid": 8486,                /* Process ID of the Message Router */
  "num_svcs": 2,             /* The number of services it provides */
  "curr_clnts": 0,          /* The number of currently connected clients */
  "tot_clnts": 3487,        /* The total number of clients connected this ses
sion */
  "total_msgs": 10635,      /* The total number of messages routed */
  "svcs": [                 /* Per service statistics */
  {
    "name": "session authenticate", /* The name of the service */
    "last_active": "2018-10-25 10:13:14", /* Last event seen by that service */
    "restarts": 1,          /* The number of process restarts */
    "pid": 8487,           /* The process ID of the Service Router */
    "queue_len": 0,        /* The current number of messages in the queue */
    "maxq_len": 200,       /* The maximum number of entries in the queue */
    "replies": 0,          /* (not applicable for authentication service) */
    "maxq_sz": 0,          /* The maximum queue size in bytes */
    "requests": 3487      /* The number of messages routed by this service
*/
  },
  {
    "name": "event log",    /* The name of the service */
    "last_active": "2018-10-25 10:13:01", /* Last event seen by that service */
    "restarts": 1,          /* The number of process restarts */
    "pid": 8488,           /* The process ID of the Service Router */
    "queue_len": 0,        /* The current number of messages in the queue */
    "maxq_len": 200,       /* The maximum number of entries in the queue */
    "replies": 0,          /* The number of replies to services */
    "maxq_sz": 102408,     /* The maximum queue size in bytes */
    "requests": 0          /* The number of messages routed by this service
*/
  } ]
}
```

If the error **6101.53 Error accessing Message Router statistics - No such file or directory** is displayed, the Message Router is not running, and should be restarted to continue normal operations.



Note: The setting `restservice <yes|no>` can be configured in `pb.settings` so that `pblighttpd-svc` does not start up the REST service. However, it allows the Message Router to run.

If the Message Router is not running and messages need to be routed to the various services, the requests are queued in the Message Router directory (normally `/opt/pbul/msgrouter`) and are similar to those in the example:



Example:

```
# ls -l /opt/pbul/msgrouter/
total 100
-rwx----- 1 root root  5482 Oct 25 10:26 wq_0004
-rwx----- 1 root root 10251 Oct 25 10:26 wq_0255
-rwx----- 1 root root 10251 Oct 25 10:26 wq_0501
-rwx----- 1 root root  5482 Oct 25 10:26 wq_0550
-rwx----- 1 root root 10251 Oct 25 10:26 wq_0684
-rwx----- 1 root root  5482 Oct 25 10:26 wq_0755
-rwx----- 1 root root 10251 Oct 25 10:26 wq_0785
-rwx----- 1 root root  5482 Oct 25 10:26 wq_0858
-rwx----- 1 root root  5482 Oct 25 10:26 wq_0869
-rwx----- 1 root root 10251 Oct 25 10:26 wq_0912
```

These Message Router Write Queue files are used for temporary storage when the Message Router is unavailable or under severe load. Once the Message Router is available again, it consumes these files and stores the data in the appropriate databases.



Note: This presents a significant difference from previous Privilege Management for Unix and Linux functionality in that event logs and other integrated products are not updated while the Message Router is unavailable. The data is stored securely in the Message Router Write Queues until it can be processed in the normal manner when the Message Router is available again.

The Message Router logs all errors in the REST log, in the specified directory, and so we recommend that this log be regularly monitored. One of the warnings that may be displayed is `WARNING: Out of free slots for the Message Router. Consider increasing 'messengerouterqueuesize' to avoid slowdown.`

This warning is not critical and is simply stating that the Message Router is under an increased load and would run faster if the specific setting is increased. If your system is displaying this warning on a regular basis, and often experiences heavy load, we recommend that you increase the `messengerouterqueuesize` setting, incrementing by roughly 25% each time. This does, however, use up more memory resources on the system as larger shared memory queues are used.



Note: `pbconfigd` has a `--call` option. This action requires a JSON string parameter to process and processes as if the call was made over REST. This allows specific calls that are required to action licensing and message router queuing calls to be made.



For more information, please see the following:

- For instructions on restarting the `pblighttpd` service, "[pblighttpd Service](#)" on page 515
- "[messengerouterqueuesize](#)" on page 126

Firewalls

This section discusses advanced configuration options for firewalls.

Privilege Management for Unix and Linux can communicate through firewalls. To configure a firewall, it is necessary to know the following:

- TCP/IP concepts and terms
- How Privilege Management for Unix and Linux establishes a connection
- Which directions the firewall is filtering

TCP/IP Concepts and Terms

In a TCP/IP network, a connection or communications channel between two processes is made up of a path through the network with a socket at each end. Each socket has its own identifying address consisting of its host ID (IP address) and a port number. The combination of the two socket addresses uniquely defines the channel in that network.

Port numbers can be assigned by the user or selected by the operating system. In some cases, the port number is well known and established by tradition. For example, FTP has used port 21 for many years. To establish an FTP session, a process opens a socket (identified by the local host ID and a port number that is assigned by the operating system) and attempts to connect it to the socket that is identified by the combination of the remote host ID and port 21. Historically, port numbers below 1024 are usually reserved for programs that are run by root, while ports from 1024 through 65535 are open for use by any program.

By default, **pbmasterd** uses port **24345**, **pblockd** uses port **24346**, **pblogd** uses port **24347**, and **pbguid** uses ports **24348** and/or **24349**. The user can reassign these by changing the port number in their services and in the settings file. Aside from those ports, Privilege Management for Unix and Linux is content to let the operating systems assign ports for the rest of its connections, unless there is a firewall in the middle.

A firewall is a security mechanism that controls network traffic that tries to pass through it. Privilege Management for Unix and Linux can work with packet-filtering firewalls. A packet-filtering firewall can allow traffic on designated ports to pass through it with no filtering.

To work with a firewall, both Privilege Management for Unix and Linux and the firewall have to agree on which ports can pass through the firewall. As an extra measure of security, Privilege Management for Unix and Linux makes use of reserved ports (numbered less than 1024) to initiate connections across a firewall. Doing so helps assure the remote machine that it is being contacted by a **root**-enabled program on the other end. The remote end of the connection can use any non-reserved ports that the firewall permits.

Previously, connections could be originated using reserved ports only. Beginning with v3.2, connections can be originated using reserved and non-reserved ports. On the receiving side, connections from non-reserved ports can be enabled by setting **allownonreservedconnections** to **true**.

The range of reserved ports that can be used to initiate outbound connections is defined in the settings file in v3.2 and later using **minoutgoingport** and **maxoutgoingport**.

In v3.2 and later, the non-reserved ports that can be used on the receiving side are defined in the settings file using **minlisteningport** and **maxlisteningport**.



For more information, please see the following:

- *"[allownonreservedconnections](#)" on page 110*
- *"[minoutgoingport and maxoutgoingport](#)" on page 111*
- *"[minlisteningport and maxlisteningport](#)" on page 110*

Privilege Management for Unix and Linux Connections

Before you learn how Privilege Management for Unix and Linux establishes connections across a firewall, it is important to understand how Privilege Management for Unix and Linux establishes connections in general. The following example shows a typical session without I/O logging.

A user starts **pbrun** to access a command.

pbrun opens a port in the range from the **minoutgoingport** to the **maxoutgoingport** and attempts to connect to the **pbmasterd** well-known port on the policy server host.

If the **pbmasterd** policies reject the request, then **pbmasterd** opens a port in the **minoutgoingport** to **maxoutgoingport** range, and attempt to connect it to **pblogd**'s well-known port on the log host to log the rejection, and the process ends here.

If the **pbmasterd** policies accept the request, then **pbmasterd** opens a port in the **minoutgoingport** to **maxoutgoingport** range, and attempts to connect it to the **pblocald** well-known port on the run host.

pblocald opens a port in the **minoutgoingport** to **maxoutgoingport** range, and attempts to connect it to **pblogd**'s well-known port on the log host. **pblocald** then sends the accept information to the log server. **pblocald** then closes the log server connection.

If the configuration permits, **pblocald** attempts to connect directly to **pbrun**, freeing **pbmasterd** to exit from the job stream. **pblocald** does this by obtaining a port in the **minoutgoingport** to **maxoutgoingport** range and attempting to connect to a port in the **minlisteningport** to **maxlisteningport** range, which **pbrun** is listening to (**pbrun**'s actual port number is passed to **pblocald** through **pbmasterd**).

pbmasterd can then exit. If **pblocaldreconnection** is set to **true**, then **pbrun** will originate and **pblocald** will listen.

pblocald runs the requested job. When it finishes, it opens another connection to the log server to log the finish status of the job.

Connections Across a Firewall

A full Privilege Management for Unix and Linux session requires two to six connections. Generally, there are two types of connections:

- Connection to a well-known port
- Connection to a dynamic connection

When connecting to a well-known port, the originator asks the operating system for a port in the range between **minoutgoingport** and **maxoutgoingport**. The port number is selected by the operating system and is called an *ephemeral port*. This port is opened on the originating side and connected to the well-known port on the target side. The firewall must be able to pass traffic on the well-known port. No configuration is needed for Privilege Management for Unix and Linux in this case. The following table summarizes the connection information for well-known ports.

Originator	Outbound Port Type	Target	Inbound Port Type	Default
pbrun	Ephemeral	pbmasterd	Well known	24345
pbmastered	Ephemeral	pblocald	Well known	24346
pbmasterd	Ephemeral	pblogd	Well known	24347
pblocald	Ephemeral	pblogd	Well known	24347

The second type of connection is a dynamic connection. These connections originate on a port in the range from **minoutgoingport** and **maxoutgoingport** that is selected by the originating machine's operating system (ephemeral port) and connect to a listening port on the target end (also an ephemeral port).

For this configuration, the firewall must be configured to pass a range of ports and Privilege Management for Unix and Linux must be configured to use those ports. If the originating side is filtered, the firewall needs to allow a range of reserved ports to pass. Beginning with Privilege Management for Unix and Linux v3.2, the port range should be configured in the firewall and the same range of ports should be set in **pblocald**'s settings file with the settings **minlisteningport** and **maxlisteningport**.

If the target machine is filtering incoming traffic, then the firewall should be configured to pass the listening ports, and the settings file on the submit host and the log host should set the same port range in their respective settings files, using the **minlisteningport** and **maxlisteningport** settings (Privilege Management for Unix and Linux v3.2 and later).

The dynamic connections from **pblocald** are summarized in the following table:



Note: Setting **pbrunreconnection** and/or **pblogreconnection** to **true** reverses the direction of the connections that are listed in the following table.

Originator	Outbound Port Type	Settings File	Target	Inbound Port Type	Settings File
pblocald	Ephemeral	minoutgoingport	pbrun	Ephemeral	minlisteningport
	reserved	maxoutgoingport		non- reserved	maxlisteningport
pblocald (when logmktemp() is used in a policy)	Ephemeral	minoutgoingport	pblogd	Ephemeral	minlisteningport
	reserved	maxoutgoingport		non- reserved	maxlisteningport

Privilege Management for Unix and Linux Shells

- **Version 3.5 and earlier:** Privilege Management for Unix and Linux: shells not available.
- **Version 4.0 and later:** Privilege Management for Unix and Linux shells available.

The Privilege Management for Unix and Linux shells provide transparent command line access to Privilege Management for Unix and Linux without the use of **pbrun**. These shells are based on the public-domain Korn shell and provide the following features:

- Bourne (**pbsh**) and Korn (**pbksh**) variants: The Korn shell variant offers a convenient user interface that includes command history, command editing, and so on. The Bourne shell variant is optimized for shell scripts, size, and performance by leaving out the extra user-interface layer.
- Transparent authorization for every command, redirection, and built-in command.
- Control of shell scripts.
- I/O logging for the entire shell session or for selective commands.
- Event logging for every command, redirection, and built-in command.
- No need for wrapped shells.
- Native-**root** mode for maintenance.
- Enabling debug trace logging.

Beginning with Privilege Management for Unix and Linux 5.0, the shells **pbsh** and **pbksh** honor the **runuser**, **rungroup**, and **runumask** settings for built-ins and I/O redirection. This feature enables built-ins to run with elevated privileges without having to elevate the privileges of the entire shell. In addition, files that are created with I/O redirection have their ownership and permissions set according to these variables.

Processing

When a shell starts, it first contacts a Policy Server host to establish its initial environment. After the initial environment is established, the Policy Server host is contacted for every command, shell built-in, and shell redirection, depending on the user, the environment, and the shell settings that are described in this section.

Shell Startup

At startup time, the shell contacts a policy server host to establish the initial environment. After the initial environment is set, it does not change for the life of that shell. Shell start is sent to the policy server host with the following characteristics:

- **localmode** is set to **true**; startup is internal to the shell, and must be run in local mode.
- **pbclientmode** is set to **shell start**.
- Command contains the name of the shell.
- **argv** contains the arguments to the shell, including the shell name.

The policy configuration language can determine that this is a shell startup by testing the variable **pbclientmode** for the value **shell start**. The shell startup is always run in local mode. The policy can control the shell startup through several variables:

- **shellallowedcommands**: A list of commands that the shell can execute without consulting a policy server and without logging. Shell pattern-matching characters are allowed. Judicious use of this list can reduce processing and network overhead. The default is an empty list, which makes the shell consult a policy server host for every command.

- **shellforbiddencommands:** A list of commands that the shell should reject without consulting a policy server and without logging. Shell pattern-matching characters are allowed. Judicious use of this list can reduce processing and network overhead. The default is an empty list.
- **shellcheckbuiltins:** When set to **true**, the shell consults a policy server host for every shell built-in command as if it were a command. The default is **false**.
- **shellcheckredirections:** When set to **true**, the shell consults a policy server host for every redirection attempt as if it were a command. The default is **false**. The shell redirections that the shells can check include `<`, `>` and `>>`.
- **shelllogincludefiles:** When set to **true**, the shell logs I/O streams for included (sourced) shell scripts, such as `.profile`. The default is **false**.
- **shellreadonly:** A list of environment variables that are made read-only in the shell session. The default is an empty list.
- **shellrestricted:** When set to **true**, the shell is run as a restricted shell using the shell's native restrictions. The default is **false**. These native restrictions include:
 - The change directory built-in command, `cd`, is disabled.
 - The environment variables **SHELL**, **ENV**, and **PATH** are set to read-only.
 - Command names do not allow absolute or relative paths.
 - Redirections that create files are not allowed (`>`, `>>`, `<>` and `>|`).
 - The **iolog** setting contains the full path of an I/O log file that records the entire shell session.

Shell Commands

Every command that the shell processes, except those that are in the **shellallowedcommands** or **shellforbiddencommands** lists, are authenticated by a policy server host. The commands are sent to the policy server host with the following characteristics:

- **localmode** is set to **true**: The policy can override this setting by setting **runlocalmode** to **false**.
- **pbclientmode** is set to **shell command**.
- **command** contains the full path of the command.
- **argv** contains the arguments to the command, including the program name.

If you want to log the I/O streams for a specific command, then you can set the **iolog** variable to an I/O log file name, and the set **runlocalmode** variable to **false**.



Note: If the **iolog** variable is set to an I/O log file name but a log server is not accessible or is not specified in the **logservers** setting, then the I/O log is stored locally on the machine where the shell is invoked.

If **shellallowedcommands** and **shellforbiddencommands** are set, then commands are examined with the following precedence:

1. Commands that exactly match (no shell pattern-matching characters) in the **shellallowedcommands** list are accepted.
2. Commands that exactly match (no shell pattern-matching characters) in the **shellforbiddencommands** list are rejected.
3. Commands that match templates (with shell pattern-matching characters) in the **shellallowedcommands** list are accepted.
4. Commands that match templates (with shell pattern-matching characters) in the **shellforbiddencommands** list are rejected.
5. All other commands are sent to a policy server host.

Shell Built-in Commands

All shells contain built-in commands, that is, commands that are part of the shell rather than separate programs. The built-in commands include:

.	exit	pwd	true
:	export	read	typeset
bg	false	readonly	ulimit
bind	fc	return	umask
break	fg	set	unalias
builtin	getopts	shift	unset
cd	hash	test	wait
continue	jobs	[and]	whence
echo	kill	time	
eval	let	times	
exec	print	trap	

When **shellcheckbuiltins** is set to **true**, the built-in commands are sent to a policy server host with the following characteristics:

- **localmode** is set to **true**; built-in commands are internal to the shell, and must be run in local mode.
- **pbclientmode** is set to **shell builtin**.
- Command contains the name of the built-in command.
- **argv** contains the arguments to the built-in command, including the built-in command name.

Shell Redirections

All shells enable I/O redirection. When **shellcheckredirections** is set to **true**, any redirection that reads or writes a file is sent to a policy server host with the following characteristics:

- **localmode** is set to **true**; redirections are internal to the shell, and must be run in local mode.
- **pbclientmode** is set to **shell redirect**.
- **argv[0]** contains **shell redirect read**, **shell redirect write**, or "".
- **argv[1]** contains the name of the file.

Shell Script Processing

When a shell executes an authorized script file, it attempts to execute it in the following sequence:

1. If the script starts with an interpreter line (**#!<program name>**), then the shell executes the script with the named program.
2. If there is no interpreter program, then the shell looks for the environment variable **EXECShell** and executes the script with that value.
3. If there is no **EXECShell**, then the shell looks for the environment variable **SHELL** and executes the script with that value.
4. If there is no **SHELL**, then the shell executes the program with **/bin/sh**.

If the script is executed by one of the Privilege Management for Unix and Linux shells, then all of the commands within the script are controlled by Privilege Management for Unix and Linux. If any other program executes the script, then the script is run in its entirety with no further control from Privilege Management for Unix and Linux.

Native Root Mode

All machines experience failures from time to time. When a network fails or no local policy server host is available, root may need to do maintenance, reboot the machine, and so on. In this case, the Privilege Management for Unix and Linux shells, when started by someone logged in as **root**, allow that user to perform the needed maintenance with minimal interference.

In this mode, the shells allow the native **root** user to perform all commands. It attempts to record events for all commands, built-in commands, and redirections, using a log server. Similarly, the shells attempt to log the I/O streams for the native root session. If the shell cannot contact a log server, then it logs both the events and the keystrokes to the local file system.

Enable Debug Trace Logging for pbsh and pbksh

You can enable debug trace logging for **pbsh** and **pbksh**, and the other Privilege Management for Unix and Linux components that participate in processing the shell commands.

To enable tracing for **pbksh**:

```
pbksh --debug=<level number>
```

The debug trace log can be found in same directory specified in the **kshlog** setting. To enable tracing for **pbsh**:

```
pbsh --debug=<level number>
```

The debug trace log can be found in same directory specified in the **shlog** setting.



For more information, please see "[Debug Trace Logging](#)" on page 330.

System Upgrades

Operating system, hardware, and network upgrades can affect your installation. Before making any changes to your system, contact a BeyondTrust Technical Support representative to review any corresponding changes that might be needed for your installation. We also strongly recommend that you run **pbbench** before and after making system changes. If you are making changes to a policy server host, then we also recommend **pbcheck**.

Operating System and Hardware Upgrades

Privilege Management for Unix and Linux must be reinstalled when making operating system or hardware upgrades. In the case of hardware upgrades, it may also be necessary to request a new license string from your BeyondTrust sales representative. If the unique machine ID generated by **pbadm** **--info --uid** for a primary license server host changes, then your current license string becomes invalid and Privilege Management for Unix and Linux no longer works.

Network Upgrades

Changes to the network environment may make it necessary to change the Privilege Management for Unix and Linux configuration. The specific actions required to configure Privilege Management for Unix and Linux during a network upgrade vary with the items that are installed or modified.

Add NIS, NIS+, and Netgroups

The NIS services maps must be updated if NIS, NIS+, or netgroups is configured after Privilege Management for Unix and Linux installation. Any superdaemons (**inetd** or **xinetd**) must also be restarted so that the new or updated port information is read from NIS.

Add Kerberos

If Kerberos is configured after Privilege Management for Unix and Linux installation, the Kerberos settings in the settings file need to be updated:

- Change the existing **kerberos** setting from **no** to **yes**.
- Check the **keytab** setting to see if it is relevant.
- Ensure that all of the principals and other Kerberos artifacts are added as appropriate.



For more information, please see "[keytab](#)" on page 244.

Add NFS

When mounting file systems across different platforms, ensure that the correct platform executable files are available for each platform and that the **PATH** environment variable points to the correct executable files for that platform.

Remember that Privilege Management for Unix and Linux daemon error logs, event logs, and I/O logs on remote mounted file systems are not supported.

Add DNS

When using DNS to resolve unqualified host names, run **nslookup** to verify a host name exists and resolves properly.

In all cases, remember to do the name and reverse lookup. Both must work correctly for Privilege Management for Unix and Linux to function.

Add a Firewall

If your installation needs to be updated to work through a firewall, then specific entries need to be updated or added in the **/etc/pb.settings** file.

Add SSL

Entries need to be updated or added in the **/etc/pb.settings** file if your Privilege Management for Unix and Linux installation needs to be updated to work with SSL.

Administration Programs

This section describes the Privilege Management for Unix and Linux system administration programs and their options.



For detailed information about installation-related programs, including package installation, please see the [Privilege Management for Unix and Linux Installation Guide](https://www.beyondtrust.com/docs/privilege-management/unix-linux/index.htm) at <https://www.beyondtrust.com/docs/privilege-management/unix-linux/index.htm>.

pbbench

The **pbbench** program tests installation and network configuration. If **pbbench** detects an error, an error message is printed to **stdout**. If no errors are detected, **pbbench** returns silently. **pbbench** generates a report that includes information about the tests that were performed, the results of the tests, and any errors that were encountered.

pbbench checks for very old versions of Privilege Management for Unix and Linux (prior to v2.0) by looking for **/etc/pb.ports** and **/etc/pb.masters** and reports a warning if these are found. The HTML GUI version of **pbbench** does not check for the Privilege Management for Unix and Linux pre-v2 files.

You can use the **--skip-*** options (such as **--skip-inetd** and **--skip-logs**) to prevent **pbbench** from performing those tests. However, the **--skip-logs**, **--skip-gui**, and **--skip-sync** options do not prevent **pbbench** from testing the connections to those hosts. To suppress the connection tests to these hosts, you must specify the **--skip-connect** option as well. This option suppresses connection tests for all host types.

To conduct connection tests only, use the **-l** (for log host connection tests) or **-m** (for policy server host connection tests) options. These options can be specified individually or together, for a single specified host name or IP address, or for all configured log hosts or policy server hosts. For both of these options, current output messages are skipped, and a single message is issued to **stdout** containing the version of the connected Privilege Management for Unix and Linux daemon (or **connection failed**). The exit status is zero if the specified host (or every configured policy server/log host) is successfully contacted. The exit status is non-zero if any policy server/log host cannot be contacted.

Syntax

- **Version 3.5 and earlier:** long command options not supported.
- **Version 4.0 and later:** long command options supported.

```
pbbench [options]
-e, --stderr
-E, --errors
-l, --logServerTest=[host_name|IP_address|SRV lookup|`external program`]
-m, --masterServerTest=[host_name|IP_address|SRV lookup|`external program`]
-V, --verbose
    --no-timeouts
    --skip-connect
    --skip-inetd
    --skip-old
    --skip-logs
    --skip-gui
    --skip-path
    --skip-shells
```

```

--skip-sync
pbbench -v | --version
pbbench --help
    
```

Arguments

-e, --stderr	Optional. Send all output to stderr .
-E, --error	Optional. Treat warnings as errors.
-l, --logServerTest =[host_name IP_address SRV lookup "external program"]	<p>Optional. Bypasses all existing tests and performs only the log host connection test on the specified host_name or IP_address. If the host_name or IP_address is not specified, the connection test is performed for all configured log hosts.</p> <p>Successful connections reported when running pbbench -l from the policy server host tell you that the log server is available, and Accept/Reject events can be logged when optimized run mode is used.</p> <p>Successful connections reported when running pbbench -l from the run host tell you if Accepts/Finish events can be logged via pblocald, and pblocald can perform I/O logging.</p> <p>Successful connections reported when running pbbench -l from the submit host tell you that Finish events can be logged while in optimized run mode and I/O logging is possible while in optimized run mode.</p> <p>Version 6.2 and earlier: option not available .</p> <p>Version 7.0 and later: option available .</p>
-m, --masterServerTest =[host_name IP_address SRV lookup "external program"]	<p>Optional. Bypasses all existing tests and performs only the policy server host connection test on the specified host_name or IP_address. If the host_name or IP_address is not specified, the connection test is performed for all configured policy server hosts.</p> <p>Version 6.2 and earlier: option not available</p> <p>Version 7.0 and later: option available</p>
-V, --verbose	Optional. Verbose mode.
--no-timeouts	Optional. Disable connection test timeouts.
--skip-connect	Optional. Disable connection tests for all host types.
--skip-inetd	Optional. Disable superdaemon (inetd/xinetd) tests.
--skip-old	Optional. Disable checking of old constructs.
--skip-logs	Optional. Disable log file tests.
--skip-gui	Optional. Disable pbguid -related tests.
--skip-path	Disable path tests of executable files.
--skip-shells	Disable tests of /etc/shells .

--skip-sync	Optional. Disable pbsync -related tests. Version 4.0 and earlier: option not available. Version 5.0 and later: option available.
-v, --version	Optional. Display the pbbench version and exit.
--help	Optional. Display program help and usage information.

Files

- `/etc/inetd.conf`
- `/opt/pbul/policies/pb.conf`
- `/etc/pb.key`
- `/etc/pb.masters`
- `/etc/pb.ports`
- `/etc/pb.settings`
- `/etc/pbmasters`
- `/etc/resolv.conf`
- `/etc/services`
- `/etc/syslog.conf`
- `/etc/xinetd.conf` DNS
- `mc NIS NIS+`
- `PBexternal pbrun`
- `SMF`
 - **Version 3.5 and earlier:** Solaris SMF not supported.
 - **Version 3.5.4 and later:** Solaris SMF supported.



Example: Run **pbbench** redirect output to the file **pbbench.output** rather than standard error:

```
pbbench > pbbench.output
```



For more information, please see the following:

- ["pbcheck" on page 371](#)
- ["pbguid" on page 422](#)
- ["pbsync" on page 473](#)
- ["pbsyncd" on page 476](#)

pbcall

The **pbcall** program is used for two purposes. To test, on the local machine, what a given function call would do, and to allow a Privilege Management for Unix and Linux policy language function to be executed from the command line.

Syntax

```
pbcall -policy-function-name args ...
```

Arguments

-policy-function-name	Required. The name of the policy function.
args	Optional. The arguments to the function.

Files

None



Example: Using the Privilege Management for Unix and Linux **stat()** function:

```
pbcall -stat /etc
```

This command returns, on the screen, the results of a call to the Privilege Management for Unix and Linux **stat** call. Using this program from a Privilege Management for Unix and Linux session, you can execute a command such as:

```
list = system("/usr/local/bin/pbrun -h "+submitHost+
"/usr/local/bin/pbcall -stat /etc");
statresult = split(list, ",");
```

This command gives the same results as:

```
statresult = stat("/etc")
```

except that it is executed on the local machine as opposed to the policy server host.

Multiple Privilege Management for Unix and Linux functions can be called at the one time:

```
pbcall -gethome user1 -getname user1
```

The output is put on different lines.

Lists need to be specified within quotation marks. For example:

```
pbcall -search "{a,b,c}" a
```



Strings can be quoted using single quotation marks or double quotation marks.

pbcheck

The **pbcheck** program processes a policy file and produces a report of syntax or language problems, or an entitlement report. The policy server daemon (**pbmasterd**) reports any policy file errors to a log file; however, **pbcheck** should be used to detect errors before you install a policy file on a live system.

With no options, **pbcheck** performs a run-check on the policy file that is specified in your settings file.



Note: The **-c** option to check licenses has been retired. Use **pbadmin --lic -l** for this purpose.

With the **-e** option, **pbcheck** produces an entitlement report in CSV format. The output data contains the columns **submithost**, **user**, **command**, **argv**, **runhost**, **runuser**, **runcommand**, and **runargv**. When a field has no value, the entitlement report displays an empty string for that field.

The policy file must first be syntactically correct, so you should run **pbcheck** to check the policy syntax before running **pbcheck -e**. Because **pbcheck -e** processes the policy without arguments, you should ensure that the logic of the policy works even if no argument is passed to it.

With the **-U**, **-C**, or **-H** options, **pbcheck** produces a formatted text entitlement report that is sorted by **user**, **command**, or **host**, respectively. If more than one of these options is specified, then each of the reports is produced, one after the other. Using the **-x** or **--csv2** option, you can export the entitlement data to Microsoft® Excel CSV format.

In CSV format, the resulting data is presented as comma-separated values and contains ASCII-formatted information for the following (in this order):

- Submit host
- User
- Command
- Argv
- Accept/reject/error text
- Run host
- Run user
- Run command
- Run argv
- **lolog (yes/no):** Not displayed when detail level is low
- **Policy file name:** Not displayed when detail level is low
- **Policy line number:** Not displayed when detail level is low
- **Policy Server host:** Displayed only when detail level is high
- **Dependencies:** Displayed only when detail level is high
- **Constraints:** Semi-colon separated. Displayed only when detail level is high

The fields displayed in this format are displayed as empty quotation marks ("") if they were not specified in the policy; this means they could be any value.

If a field contains *Can not evaluate soft ... expression* in the output, it means the policy was setting the corresponding variable to a value that can be evaluated only at runtime (when running **pbrun**), such as **argv**, **date**, **time**, and so on.

With the option **-H**, each submit host is displayed on the top, followed by as many lines as necessary containing the other fields in the same order as above (except the submit host field).


Example:

```
Host: <submitHost>
<submitUser> <command> <argv> <Accept/Reject/Error Text> <runHost> <runUser>
<runCommand> <runArgv> <iolog> <policyName> <policyLineNumber> <masterHost>
<dependency> <constraints>
```

With the option **-U**, each user is displayed on the top, followed by as many lines as necessary containing the other fields in the same order as above (except the user field).


Example:

```
User: <user>
<submitHost> <command> <argv> <Accept/Reject/Error Text> <runHost> <runUser>
<runCommand> <runArgv> <iolog> <policyName> <policyLineNumber> <masterHost>
<dependency> <constraints>
```

With the option **-C**, each command is displayed on the top, followed by as many lines as necessary containing the other fields in the same order as above (except the command field).


Example:

```
Command: <command>
<submitHost> <submitUser> <argv> <Accept/Reject/Error Text> <runHost>
<runUser> <runCommand> <runArgv> <iolog> <policyName> <policyLineNumber>
<masterHost> <dependency> <constraints>
```

The fields displayed in **-H**, **-U**, and **-C** are displayed as **<UNSPECIFIED>** if they were not specified in the policy, and means they could have any value.

The fields **iolog**, **polycname**, and **policyLineNumber** are not displayed if detail level is low, and the fields **masterhost**, **dependency**, and **constraints** are displayed only when detail level is high.

The entitlement report mechanism uses a temporary work file before creating the final report. This temporary file is created in the **\$TMPDIR** directory. If the **TMPDIR** environment variable is not set (or is set to **/tmp/**), the workfile is written in the **/tmp** directory.



Note: Temporary files that are owned by **root** in **/tmp** are a potential vulnerability. We recommend you set **TMPDIR=/var** or some other appropriate directory where normal users do not have write access.

Policies that use looping constructs which modify the iterating variable within the loop do not work correctly. Policies that test the contents of **argv** or **runargv** may produce incomplete results.

Policies that use data that is known only at an actual run-time (such as the **date** and **argv** variables) produce incomplete results. In these cases, warnings and errors use the term *soft* when referring to variables that cannot be fully evaluated.

If a policy contains conditions that are based on external **data** sources or on external files that are generated at run time, the entitlements are evaluated as if the conditions are true. The last column (displayed when the **--detail** option is set to **High**), **Constraint**, shows that each row is displayed when a condition is accepted. For example, if the policy checks whether a user belongs to a certain netgroup before

accepting the command, the report shows the command for this user as accepted; the **Constraint** field shows the condition that the user belongs to the netgroup. At runtime, a particular user listed in the report might actually not be part of that netgroup, and therefore gets rejected.

The constraint option enables you to limit the report by using any valid Privilege Management for Unix and Linux policy language expression.


Example:

```
-c "user=='dba'" -c "host in {'A', 'B'}"
```

*This limits the report to **dba** privileges on hosts **A** and **B**.*

Most user defined functions are not properly processed for entitlement reporting.

pbcheck evaluates a limited set of functions in a constraint when those functions are used against the output variables **submithost**, **user**, **command**, **runhost**, **runuser**, and **runcommand**. The supported functions are **basename()**, **tolower()**, and **toupper()**.

Examples of constraints whose function calls are evaluated are:

- **basename(command)** in **Allowed_Cmds**
- **tolower(user)** in **Allowed_Users**
- **toupper(submithost)** in **Allowed_Hosts**

All other functions are not evaluated, and do not show actual values for the affected output field.



Example: *The following policy does not produce specific values in the command field:*

```
gsub("^.*/", "", command) in Allowed_Cmds
```

It does, however, insert that whole constraint in the command field for informational purposes.



Note: *The **-b** (**--nobasename**) option for **pbcheck** has been deprecated because **pbcheck** now evaluates the **basename()** function.*

Syntax

- **Version 3.5 and earlier:** long command options not supported.
- **Version 4.0 and later:** long command options supported.


```
pbcheck [options] [command [arguments]]
-f, --file=file_name
-h, --host=host_name
-r, --run
-s, --syntax
-t, --type
-x, --csv2
```

```

-p, --policydir=directory
-L, --showlists[=listname [,listname]...]
-S, --showduplicates
pbcheck [options] [-c constraint [-c constraint]...] [command [arguments]]
-f, --file=file_name
-p, --policydir=directory
-e, --entitlement [=abridged|standard|extended| exhaustive]
-b, --nobasename
-l, --uselistnames [=columnname [,columnname]...]
--maxchildren <number>
--maxloopchildren <number>
-U, --user_report
-H, --host_report
-C, --command_report
-D, --detail [=low|medium|high]
-d, --display_headers
-A, --accepted
-R, --rejected
-I, --rejected_implicit
-c, --constraint=expression
-x, --csv2
pbcheck -v | --version
pbcheck --help
    
```

Arguments

-A, --accepted	<p>Optional. For entitlement reporting only. Report accepted commands (default).</p> <p>May be combined with -I and -R.</p> <p>Version 4.0 and earlier: option not available.</p> <p>Version 5.0 and later: option available.</p>
-b, --nobasename	<p>Obsolete. Optional. For entitlement reporting only. Remove basename function calls from the entitlement report, leaving only the command or runcommand.</p> <p>Version 6.0 and earlier: option not available.</p> <p>Version 6.1 through 6.2.2: option available.</p> <p>Version 6.2.3 and later: option deprecated.</p> <p>This is the built-in default behavior of pbcheck.</p>
-c, --constraint =expression	<p>Optional. Restrict the entitlement report to items that match the expression (for example, -c "user=='root'").</p> <p>Version 4.0 and earlier: option not available.</p> <p>Version 5.0 and later: option available.</p>
-C, --command_report	<p>Optional. Produce a formatted entitlement report in command sequence.</p> <p>Version 4.0 and earlier: option not available.</p> <p>Version 5.0 and later: option available.</p>

-d, --display_headers	<p>Optional. Display the field headers as the top of an entitlement report.</p> <p>Version 6.0.0 and earlier: option not available.</p> <p>Version 6.0.1 and later: option available.</p>
-D, --detail [=low medium high]	<p>Optional. Produce the specified level of detail.</p> <p>Version 4.0 and earlier: option not available.</p> <p>Version 5.0 and later: option available.</p>
-e, --entitlement [=abridged standard extended exhaustive]	<p>Optional. Produce entitlement data in CSV format. The optional arguments are:</p> <p>abridged: Simplest and fastest method. It ignores the interactions between different if statements and groups similar data (for example, users or hosts) during evaluation.</p> <p>standard (default): Slower and more detailed method. It tracks the interactions between different conditionals.</p> <p>extended: Method providing more detail. It tracks interactions between different conditionals and evaluates the policy for each member of a list, rather than treating the list as a group.</p> <p>exhaustive: Method providing more detail. The output is currently the same as extended, but may be changed in the future.</p> <p>Version 4.0 and earlier: option not available.</p> <p>Version 5.0 and later: option available.</p>
-f, --file=file_name	<p>Optional. The file name to check. Defaults to the normal policy file.</p>
-h, --host=host_name	<p>Optional. Simulate a check with a remote host name.</p>
-H, --host_report	<p>Optional. Produce a formatted entitlement report in host sequence.</p> <p>Version 4.0 and earlier: option not available.</p> <p>Version 5.0 and later: option available.</p>
-I, --rejected_implicit	<p>Optional. For entitlement reporting only. Report commands that are implicitly rejected at the end of the policy. May be combined with -A and -R.</p> <p>Version 4.0 and earlier: option not available.</p> <p>Version 5.0 and later: option available.</p>
-l, --uselistnames [=column_ name [,column_name]...]	<p>Optional. For entitlement reporting only. Replace the list values with the name of the list variable for all fields or for specific fields. Valid field names are submitHost, submitUser, command, runHost, runUser, and runCommand. When specifying multiple field names, enter them as comma-separated values (no spaces). Omit the field name to replace all applicable fields with the list variable name.</p> <p>Statements that act on individual items (for example, switch and case) cause the list to be expanded in the report, regardless of the -l option.</p> <div data-bbox="467 1738 1515 1852" style="border: 1px solid black; padding: 5px; margin-top: 10px;"> <p> Note: If the run variables <i>runuser</i> and <i>runcommand</i> are not explicitly set in the policy, their values will depend on the variables <i>user</i> and <i>command</i>. For more information, please see the Privilege Management for Unix and Linux Policy Language Guide at</p> </div>



<https://www.beyondtrust.com/docs/privilege-management/unix-linux/index.htm>

In such a case, the **--uselistnames** option applied to **user** and **command** fields (**submitUser**, **command**) automatically affect the output of their corresponding run variables (fields **runUser** and **runCommand**). Similarly, if a policy has an explicit dependency for **runhost** on **submithost**, then the **--uselistnames** option applied to the **submitHost** field are also reflected in the **runHost** field.

Version 6.0 and earlier: option not available.

Version 6.1 and later: option available.

**-L, --showlists [listname
[,listname]...]**

Optional. Show members of all lists or of specific lists. Omit the list name to display members of all lists.

Version 6.2.2 and earlier: option not available.

Version 6.2.3 and later: option available.

--maxchildren <number>

Limit the total number of live **pbcheck** descendant processes. After this limit is reached, the entire **pbcheck** process tree is terminated. This is a safety mechanism to prevent crippling a system with too many processes. The default value is **200**. Increasing the value should only be necessary for policies with a large number of **if**, **case**, or looping statements that do not result in an Accept or a Reject.

Version 6.1 and earlier: option not available.

Version 6.2 and later: option available.

--maxloopchildren <number>

Limit the number of child processes that evaluate the same policy line (for example, an endless loop). After this limit is reached, the process that encounters the same line for the specified number of iterations issues a diagnostic message indicating the policy line and statement. Then the process exits. That child's parent and all ancestors are free to continue processing, resulting in a possibly incomplete entitlement report. This is a safety mechanism to prevent crippling a system with too many processes. The default value is **25**.

Version 6.1 and earlier: option not available.

Version 6.2 and later: option available.

-p, --policydir =directory

Optional. Provide a policy directory to control **include** files in the configuration policy. If a file name in an **include** statement starts with a slash (**/**), then that file is used. Otherwise, the directory that is specified using the **-p** option is used to search for the file in the **include** statement.

-r, --run

Optional. Perform run-time checking (forces syntax and type checking).

-R, --rejected

Optional. For entitlement reporting only. Report commands that are rejected by an explicit Reject statement. May be combined with **-A** and **-I**.

Version 4.0 and earlier: option not available.

Version 5.0 and later: option available.

-s, --syntax

Optional. Perform syntax checking.

-S, --showduplicates

Optional. Show information on lists with duplicate members.

Version 6.2.2 and earlier: option not available.

	Version 6.2.3 and later: option available.
-t, --type	Optional. Perform type checking (forces syntax checking).
-U, --user_report	Optional. Produce a formatted entitlement report in user sequence. Version 4.0 and earlier: option not available. Version 5.0 and later: option available.
[command [arguments]]	Optional. Search through the policy for the specified command and arguments to display whether it is accepted or rejected. The command and its arguments must be specified last in the pbcheck argument list. If the pbcheck argument list includes options that Accept arguments, they must be specified before command.
-v, --version	Optional. Displays the version information on stderr and exits.
-x, --csv2	Optional. Export entitlement data in Microsoft Excel CSV format. Version 6.2 and earlier: option not available. Version 6.2.1 and later: option available.
--help	Optional. Displays the program's help message and exits.

Files

Privilege Management for Unix and Linux policy file



Example: Perform a syntax check of the user-specified configuration file **pb.mainconfig** located in **/etc**

```
pbcheck -f /etc/pb.mainconfig
```



For more information, please see the following:

- ["pbkey" on page 426](#)
- ["pblocald" on page 430](#)
- ["pbmasterd" on page 441](#)
- ["pbpasswd" on page 446](#)
- ["pbprint" on page 450](#)
- ["pbreplay" on page 454](#)
- ["pbrun" on page 465](#)
- ["pbsum" on page 471](#)

pbdbutil, pbadmin

- **Version 8.5.0 and earlier:** **pbdbutil** not available.
- **Version 9.0.0 and later:** **pbdbutil** available.

Starting with version 9, Privilege Management for Unix and Linux uses database files for the storage of all the normal configuration files and scripts, plus data storage for a range of new facilities. The utility **pbdbutil** provides a command line tool to maintain all of these databases.



Note: Due to the evolving nature of the **pbdbutil** command, its name will be changing to **pbadmin** in the future. To assist in this future transition a symbolic link called **pbadmin** is now automatically created for your convenience.

The command has *global* options that are used to carry out maintenance tasks on all databases, and more specific options that allow maintenance of specified databases. Each group of database options have their own usage/help.

Usage

```
pbdbutil [<options>] [ <file> <file> ...]
```

Global Options

-y	Use cached credentials for remote functionality.
-c <files(s)>	Perform database integrity check.
-K <newkeypath> <file(s)>	(Re)encrypt the database.
-O <oldkeypath>	Specify the old database key file.
-C	Output in CSV format instead of JSON.
-P	Pretty print JSON output.

Authentication Options

--auth <options...>	Various authentication options.
-h	Help on authentication options.

Info Options

--info <options...>	Various information options.
-h	Help on info options.

License Maintenance and Statistics Options

<code>--lic <options...></code>	License Maintenance and Statistics options.
<code>-h</code>	Help on License Maintenance and Statistics options.

Setting/Configuration/Key Options

<code>--cfg <options...></code>	Specify setting/config options.
<code>-h</code>	Help on Setting/Configuration options.

Role Based Policy Options

<code>--rbp <options...></code>	Role Based Policy options.
<code>-h</code>	Help on Role Based Policy options.

Client Registration Profile Options

<code>--reg <options...></code>	Client Registration options.
<code>-h</code>	Help on Client Registration options.

Management Event Options

<code>--evt <options...></code>	Event options.
<code>-h</code>	Help on Management Event options.

REST Keystore Options

<code>--rest <options...></code>	REST keystore options.
<code>-h</code>	Help on Management REST keystore options.

Sudo Policy Database Options

<code>--sudo <options...></code>	Sudo database options.
<code>-h</code>	Help on Management sudo database options.

Registry Name Service Database Options

<code>--svc <options...></code>	Registry Name Service database options.
---------------------------------------	---

Database Synchronization Options

<code>--dbsync <options...></code>	Database Synchronization options.
--	-----------------------------------

Registry Name Service Cache Options

<code>--scache <options...></code>	Registry Name Service cache options.
--	--------------------------------------

File integrity Monitor Options

<code>--fim <options...></code>	File integrity monitor options.
---------------------------------------	---------------------------------

Event Log Cache Options

<code>--evtcache <options...></code>	Event Log Cache options.
--	--------------------------

<code>-h</code>	Help on Event Log Cache options.
-----------------	----------------------------------

IO Log Cache Options

<code>--iocache <options...></code>	IO Log Cache options.
---	-----------------------

<code>-h</code>	Help on IO Log Cache options.
-----------------	-------------------------------

IO Log Queue Options

<code>--iologidx <options...></code>	IO Log Queue options.
--	-----------------------

<code>-h</code>	Help on IO Log Queue options.
-----------------	-------------------------------

Integrated Product Options

<code>--intprod <options...></code>	Integrated Product options.
---	-----------------------------

<code>-h</code>	Help on Integrated Product options.
-----------------	-------------------------------------

Global Options

<code>--check <file(s)...></code>	Do an integrity check on the specified files. If the database(s) are encrypted it attempts to read the file using the database key specified in the pb.settings file.
<code><-c --csv></code>	By default all output messages and data is output in JSON format. This option specifies output in Comma Separated Values.
<code><-p --pretty></code>	When outputting data in JSON <i>pretty</i> print the data in a more human readable form.
<code><-K --newkeypath></code> <code>[<-O --oldkeypath>]</code> <code>[<file(s)...>]</code>	Reencrypt database file(s) using the specified new key. If the old key path is not supplied it attempts to open the database file with the key specified in pb.settings file.

Setting/Configuration Options

These options provide methods to import, maintain and export the settings, configuration and key files that were traditionally kept in files in Privilege Management for Unix and Linux. These files can now be imported into a database which provide versioning and change management, methods to retrieve, update, and save settings and configuration across the enterprise in a secure manner using the Privilege Management for Unix and Linux REST services.

These options need to be specified after the `--cfg` option.

Usage

```
pbdbutil --cfg [<options>] [ <file> <file> ...]
```



<code>-i [<file(s)>]</code>	Import/update all or specified .cfg file(s) in the database.
<code>-m <msg></code>	Specify message. Required when change management enabled.
<code>-N</code>	Do not rename file on import.
<code>-e [<files(s)>]</code>	Export all or specified .cfg file(s) in the database.
<code>-e -o <outfile></code> <code><file></code>	Export .cfg file from database and output to new file name.
<code>--force</code>	Force the overwrite of the output file when exporting.
<code>-V <ver tag></code>	Used with export .cfg file, but export given version or tag.
<code>-D [<file(s)>]</code>	Diff all/specified file(s) with current exported file(s).
<code>-V <from:to></code>	Used to specify from/to versions to diff.
<code>-V <ver tag></code>	Used to specify version or tag to diff.
<code>-r [<files(s)>]</code>	Mark specified .cfg file(s) deleted in the database.
<code>-l</code>	List all .cfg files in the database.

-s <[- +]attribute>	Sort the list of records by attribute (asc/desc).
-l <file(s)>	List version information of .cfg file(s) in the database.
-t <tag> [<file(s)>]	Tag .cfg file(s) in the database at current version.
-x <tag> [<file(s)>]	Delete tag from .cfg file(s) in the database.
-k <encryption> <file(s)>	Encrypt .cfg file(s) in the database.
n [--force] <file(s)>	Create new key file(s) in the database
-U	Force unlock of locked cfg files in the database
-A <file> <svcgroup> <...>	Set file as being automatically synchronized within Service Group
-X <file> <svcgroup> <...>	Unset file as being automatically synchronized within Service Group
-L	List synchronization configuration for cfg files in the database
-u <setting> <arg>[<argN>]	Set the setting in the current settings file.
-o <file>	Set the setting in the specified settings file. Need to use -o with -u .
-u '{"<setting>":"<val>","...}'	Set the setting using JSON format.
-g <setting> --value	Displays the value of the variable <setting> as set in /etc/pb.settings in a simple string format. Wildcards are allowed. If used, the output is in variable=value format.
-g <setting>	Get the setting from the current settings file in JSON format.
-o <file>	Get the setting from the specified settings file. -o need to be used with -g .
-g --default <setting>	Gets the default value of the variable <setting>. This is not the value in pb.settings, but the default value when the variable is not defined (or commented out) in pb.settings . The output is displayed in JSON format with all metadata.



Example: The following displays the values in JSON format of the variable **pbresttimeskew**:

```
# pbadmin --cfg --default -g pbresttimeskew
{"disabled":true,"description":"Max time skew between hosts
(sec)","default":60,"type":"number","gtype":4,"deprecated"
```

	 <pre>:false,"name":"pbresttimeskew"}</pre> <p>Wildcards are allowed. If used the output is in variable=value format.</p>
-g --default --value <setting>	<p>Displays the value of the variable in a simple string format.</p> <p>For example, the following command gets default variable value in simple string form:</p> <pre>pbadmin --cfg --default --value -g <var></pre> <p>Wildcards are allowed. If used the output is in variable=value format.</p>  <p>Example:</p> <pre>pbadmin --cfg --default -g *</pre> <pre>pbadmin --cfg --default -g pb*</pre>
-g '["<setting1>" , "<settingN>"]'	<p>Get the settings using JSON format.</p>
-d <setting>	<p>Delete the setting in the current settings file.</p>
-o <file>	<p>Delete the setting in the specified settings file. -o needs to be used with -d.</p>
-d '["<setting1>" , "<settingN>"]'	<p>Delete the settings using JSON format.</p>

Descriptions

"<-i --import>" "<-m --msg>" "<message>" "[<file(s)...>"]	<p>Import specified settings, configuration or key files into the /etc/pb.db database. If Change Management is enabled, a message must be supplied to log in the audit event database. If no files are specified on the command line, all files that already exist in the database are checked and imported if required.</p>
<-e --export> [-f] [<file (s)...>] <-e --export> [-f] <-V --version> <num tag> [<file (s)...>] <-e --export> [-f] <-o --output> <outfile> <file> <-e --export> [-f] <-V --version> <num tag> <-o --output> <outfile> <file>	<p>Export specified settings, configuration or key file(s) from the /etc/pb.db database. If no files are specified on the command line, all files that exist in the database are exported. Specific versions or tagged groups of files can be exported. If the output file(s) already exist the -f parameter must be applied to force the overwrite of the existing file.</p>

<code><-l --list></code>	List all the current files held in the /etc/pb.db database.
<code><-l --list> [-j] [<file (s)...>]</code>	List all the versions of specified files held in the /etc/pb.db database. By default this is displayed in .csv, but can be displayed in JSON using the -j option. Specify a tag for current versions of files that exist in the /etc/pb.db database. These files can then be exported as a tagged group to facilitate change sets of files.
<code><-t --tag> <tag text> [<file (s)...>]</code>	Specify a tag for current versions of files that exist in the /etc/pb.db database. These files can then be exported as a tagged group to facilitate change sets of files. If file names are not specified, all current versions are added to the tagged group.
<code><-d --deltag> <tag text> [<file(s)...>]</code>	Remove the tag from files specified. If file names are not specified, the tag is removed from all files that exist in the /etc/pb.db database.
<code><-k --encrypt> <algorithm> [<file(s)...>]</code>	Encrypt existing setting/configuration files in the /etc/pb.db database.
<code><-n --newkey> [<file(s)...>]</code>	Create a new key file in the /etc/pb.db database.

License Management Options

As of version 10.0, License Management is centralized and can be carried out on the primary license server using the command **pbadmin**.

This command line administration tool provides methods to update the license string, to list summary statistics and to retire clients to free up licenses.

All of the commands that list statistics can be run from any server that provides a service. All commands that update the database, such as updating the license itself or retiring clients, should be run on the primary license server:



Example:

```
pbadmin --lic -u '{ "PBULPolClnts":200, "SudoPolClnts":200, "RBPClnts":200, "ACAClnts":1,
"AKAClnts":0, "FIMClnts":0, "SOLRClnts":1, "Owner":"My Company Corp", "Comment":"Standard
License for My Company", "AutoRetire":7, "Recycle":7, "Expires":"2018-03-01 00:00:00",
"Terminates":"2019-03-01 00:00:00", "HostId":"7faf7681-4d42-4b69-00bfdad93b4a3dfb",
"HMAC":"UtGE3tD6qK2UwutY3GFOqodjdq30pEDAW2cKb5/OaMc=" } '
```

This command updates the installation with the license string provided by BeyondTrust to a standard license.

Usage

```
pbadmin --lic [<options>] ...
```

-u '{ param }'

Update primary license server license where the **{ param }**

	argument is the supplied JSON formatted license.
<code>-u <path></code>	Update primary license server license where <path> is the path to a file that contains the supplied JSON formatted license.
<code>-G</code>	Retrieve the license string and attributes.
<code>-l [<wildcard>] [-l]</code>	List client license usage summary. Supply an extra -l to detail service information.
<code>-l '{ ... ["fqdn" : "<wildcard>"], ["retired" : <true false>], ["updated_older" : <epoch>], ["updated_newer" : <epoch>], ["updated_older" : { "years" : n, "months" : n, "days" : n, "hours" : n } ["updated_newer" : { "years" : n, "months" : n, "days" : n, "hours" : n]</code>	Alternatively specify a filter expression to list only those clients that match the filter.
<code>-s <[- +]attribute></code>	Use -s to sort the list of records by attribute name (asc/desc).
<code>-L [<service>] [-L]</code>	List client Service License Usage summary. Specify an extra -L to detail client information.
<code>-r {"uuid" : "<uuid wildcard>" -r {"uuid" : ["<uuid wildcard>", "uuid", ...]} -r {"fqdn" : "<fqdn wildcard>" -r {"fqdn" : ["<fqdn wildcard>", "fqdn", ...]} --force</code>	Retire client(s) by UUID or FQDN. Use --force to over-ride warning message.
<code>-R</code>	Immediately refresh the license statistics from the primary license server.

Sample Commands

<code>pbadmin --lic -G</code>	Retrieves the full license string, detailing the entitlements and expiry of the license.
<code>pbadmin --lic -l</code>	Lists all of the clients that are currently licensed throughout the installation.
<code>pbadmin --lic -L</code>	Lists the summary statistics referenced by the Privilege Management for Unix and Linux service type.
<code>pbadmin --lic -l '{ "retired": true }'</code>	Lists all of the clients that are currently manually retired.
<code>pbadmin --lic -l '{ "fqdn" : "*.mydom.com" }'</code>	Lists all of the clients that have been licensed are in the

	mydom.com domain.
<code>pbadmin --lic -l '{ "updated_older" : "2018-01-01" }</code>	Lists all of the clients that were last updated before the 1st of January 2018.
<code>pbadmin --lic -l '{ "updated_older" : { "months" : 6 } }'</code>	Lists all of the clients that were last updated 6 months or more ago.
<code>pbadmin --lic -r '{ "uuid" : "7faf7681-4d42-4b69-00bfdad93b4a3dfc" }' --force</code>	Manually retires a client specified by its unique id.
<code>pbadmin --lic -r '{ "updated_older" : { "days" : 120 } }' --force</code>	Manually retires all clients that have not been updated in the last 120 days.

Authentication Credential Cache Options

These options allow users of **pbdbutil** to cache credentials to facilitate working with remote services.

Usage

```
pbdbutil --auth [<options>] [ <file> <file> ...]
```

Auth Options

<code>--login { "appid":"<appid>","appkey":"<appkey" [, "svc":"<svc>"] }</code>	Cache specified appid/appkey credential for authentication.
<code>--logout [{"key":"<key>","svc":"<svc>"}]</code>	Remove default or specified credential key from cache.
<code>-l</code>	List cached credentials.
<code>-h</code>	Help on auth options.

Information Options

These options provide various information about the current system configuration or status.

Usage

```
pbdbutil --info [<options>]
```

Info Options

<code>--fqdn [<hostname>]</code>	Get fully qualified name for this host or hostname.
--	---

<code>--sched</code>	List Scheduler tasks.
<code>--uuid</code>	Get the local hosts UUID.
<code>--msgs</code>	Retrieve the Message Router statistics.
<code>--restsvr</code>	Retrieve the REST Service statistics.
<code>-h</code>	Help on info options.

Role Based Policy Options

The Role Based Policy is held in multiple tables. Each table refers to an individual entity with attributes, and is referenced by unique entity ids. Each entity is then linked together into a role. When retrieving, updating, or deleting entities, either the **name** or **id** can be used. The command line utility **pbdbutil** with the option **--rbp** can be used to retrieve (**-g**), update (**-u**), or delete (**-d**) entities. When updating, complete entities including all its attributes need to be defined. The REST API uses the same JSON format and parameters, and use **GET**, **PUT** and **DELETE** respectively. There are also a number pseudo-attributes that allow the retrieval of lists based upon the parent grouping, these are:

- **usergrpname**: list User Lists which correspond to the specified User Group
- **hostgrpname**: list Host Lists which correspond to the specified Host Group
- **cmdgrpname**: list Command Lists which correspond to the specified Command Group
- **tmdategrpname**: list Time/Date Lists which correspond to the specified Time/Date Group
- **rolename**: list all lists which correspond to the specified Role Group

Usage

```
pbdbutil --rbp [<options>] [ <file> <file> ...]
```

-b -m <msg>	Begin Role Based Policy change transaction
-c	Commit Role Based Policy change transaction
-r	Rollback Role Based Policy change transaction
--force -m <msg>	Force Rollback of other users change transaction
-i <file>	Import Role Based Policy file in the database
-e -o <outfile>	Export Role Based Policy from database and output to file
-V <ver>	Used with export, but export specified version
-g { json param }	Get Role Based Policy database records
-u { json param }	Update Role Based Policy database records
-m <msg>	Specify message - required when change management enabled.
-d { json param }	Delete Role Based Policy database records
--force	Force deletion of dependent records in the database
-m <msg>	Specify message - required when change management enabled.
-n	Create new Role Based Policy database
-R { json param }	Report user entitlements from the database
-R	Add option to display commands
-R	Add option to display time/date restrictions

-R	Add option to display additional role options
-E { json param }	List user entitlements data from the database where { json param } is one or more of: "submituser" : "user1" Specify submit user or wildcard "submithost" : "host1" Specify submit host or wildcard "runuser" : "user1" Specify run user or wildcard "runhost" : "host1" Specify run host or wildcard "command" : "command" Specify command or wildcard
-L	List all Role Based Policy policies in the database
-t <tag>	Limit list by tag wildcard
-l	List all Role Based Policy versions in the database



Example: List all of the User Groups whose name matches *ug**

```
pbdbutil --rbp -g '{ "usergrp" : { "name" : "ug*" } }'
[{"id":1,"ug1":"name","description":"desc","disabled":0,"single":0,"type":"I","ext
info":null}]
```



Example: List the User Group whose *id=1*

```
pbdbutil -g '{ "usergrp" : { "id" : "1" } }'
[{"id":1,"ug1":"name","description":"desc","disabled":0,"single":0,"type":"I","ext
info":null}]
```

Record Entities

- **usergrp**
- **userlist**
- **hostgrp**
- **hostlist**
- **cmdgrp**
- **cmdlist**
- **tmdategrp**
- **tmdatelist**
- **role**
- **roleusers**

- roleghost
- rolecnds
- roletmdates

Entities can be listed by attributes **name** and **id**, and entity specific attribute names **rolename**, **usergrpname**, **hostgrpname**, **cmdgrpname**, **tmdategrpname**.


Example:

```
-g '{ "role" : { "name" : "*" }} ' Display all Roles
-g '{ "usergrp" : { "name" : "n*" }}' Display all User Groups which match "n*"
-g '{ "userlist" : { "name" : "usergrp1" }} ' Display group membership for usergrp by
name
-g '{ "roleusers" : { "rolename" : "role1" }}' Display list of usergrps assigned to role
-g '{ "rolehosts" : { "id" : 1 }} ' Display list of hostgrps assigned to role id 1
```

Descriptions

-b [-m <msg>]	<p>This option is mandatory if the Role Based Policy transactions are enabled.</p> <p>Before any changes can be made the administrator must begin the transaction with a suitable Change Management message. This transaction is then kept open until the same user commits or rolls back the transaction. The transaction is not visible by the live authorization process until it is committed.</p> <p>Available if the Role Based Policy Transactions are enabled.</p>
-c	Commit the current open transaction making it <i>live</i> .
-r [--force -m <msg>]	<p>Rollback the current open transaction, discarding any changes that have been made.</p> <p>Available if the Role Based Policy Transactions are enabled.</p>
-i <file>	Import Role Based Policy file in the database.
-e -o <outfile> [-V <ver>]	Export Role Based Policy from database and output to file.
-g { json param }	Retrieve and display attributes of the entities within the Role Based Policy database.
-u { json param }	Update entities and attributes within the Role Based Policy database.
-d { json param }	Delete entities within the Role Based Policy database.
-n	Create a new Role Based Policy database, as specified by the policydb keyword in the Privilege Management/ etc/pb.settings configuration file.
-R { json param }	Report user entitlements from the database.
-R	Add option to display commands.
-R	Add option to display time/date restrictions.
-R	Add option to display additional role options.

-E { json param }	List user entitlements data from the database where { json param } is one or more of: "submituser" : "user1" Specify submit user or wildcard "submithost" : "host1" Specify submit host or wildcard "runuser" : "user1" Specify run user or wildcard "runhost" : "host1" Specify run host or wildcard "command" : "command" Specify command or wildcard
-L	List all Role Based Policy policies in the database
-I	List all Role Based Policy versions in the database

User Group Examples



Example: Retrieve list of User Groups that match *ug**

```
-g '{ "usergrp" : { "name" : "ug*" } }' [{"id":1,"ug1":"name","description":"desc","disabled":0,"single":0,"type":"I","extinfo":null}]
```



Example: Retrieve list of Users in the User Group *ug1*

```
-g '{ "userlist" : { "usergrpname" : "ug1" } }' [{"id":1,"user":"root"}, {"id":1,"user":"adm*"}]
```



Example: Update User Group *ug1* with new attributes

```
-u '{ "usergrp" : { "id":1,"name":"ug1","description":"new description","disabled":0,"single":0,"type":"I","extinfo":null} }'
```



Example: Add new user to User Group *ug1*

```
-u '{ "userlist" : { "usergrpname":"ug1","user":"wheel"} }'
```



Example: To delete all users from User Group *ug1*

```
-d '{ "userlist" : { "usergrpname":"ug1"} }'
```



Example: To delete specified user from User Group **ug1**

```
-d '{ "userlist" : { "usergrpname":"ug1", "user" : "user1"} }'
```

Host Group Examples



Example: Retrieve list of Host Groups that match **hg***

```
-g '{ "hostgrp" : { "name" : "hg*" } }' [{"id":1,"hg1":"name","description":"desc","disabled":0,"type":"I","extinfo":null}]
```



Example: Retrieve list of Hosts in the Host Group **hg1**

```
-g '{ "hostlist" : { "hostgrpname" : "hg1" } }' [{"id":1,"host":"host2"}, {"id":1,"host":"*.dev.com"}]
```



Example: Update Host Group **hg1** with new attributes

```
-u '{ "hostgrp" : { "id":1,"name":"hg1","description":"new description","disabled":0,"type":"I","extinfo":null} }'
```



Example: Add new host to Host Group **hg1**

```
-u '{ "hostlist" : { "hostgrpname":"hg1","host":"host5"} }'
```



Example: To delete all hosts from Host Group **hg1**

```
-d '{ "hostlist" : { "hostgrpname":"hg1"} }'
```



Example: To delete specified host from Host Group **hg1**

```
-d '{ "hostlist" : { "hostgrpname":"hg1", "host" : "host1"} }'
```


Command Examples



Example: Retrieve list of Command Groups that match *cg**

```
-g '{ "cmdgrp" : { "name" : "cg*" } }'  
[{"id":1,"cgl":"name","description":"desc","disabled":0}  
]
```



Example: Retrieve list of Commands in the Command Group *cg1*

```
-g '{ "cmdlist" : { "cmdgrpname" : "cg1" } }'  
[{"id":1,"cmd":"rm *","rewrite":"echo $*"}, {"id":1,"cmd":"/usr/bin/rm  
*","rewrite":"echo $*"}]
```



Example: Update Command Group *cg1* with new attributes

```
-u '{ "cmdgrp" : { "id":1,"name":"cg1","description":"new description","disabled":0} }'
```



Example: Add new command to Command Group *cg1*

```
-u '{ "cmdlist" : { "cmdgrpname":"cg1","cmd":"/bin/rm *","rewrite":"echo  
$*"} }'
```



Example: To delete all commands from Command Group *cg1*

```
-d '{ "cmdlist" : { "cmdgrpname":"cg1"} }'
```



Example: To delete specified cmd from Command Group *cg1*

```
-d '{ "cmdlist" : { "cmdgrpname":"cg1", "cmd" : "rm *"} }'
```

Time/Date Examples



Example: Retrieve list of Time/Date Groups that match **td***

```
-g '{ "tmdategrp" : { "name" : "td*" } }'
[{"id":1,"tdl":"name","description":"desc","disabled":0}
]
```



Example: Retrieve list of Time/Dates in the Time/Date Group **td1**

```
-g '{ "tmdate" : { "tmdategrpname" : "td1" } }'
[{"id":1,"tmdate" : "{
  \"mon\" : [0,0,0,0,0,0,0,15,15,15,15,15,15,15,15,15,15,15,15,3,0,0,0,0,0,0],
  \"tue\" : [0,0,0,0,0,0,0,15,15,15,15,15,15,15,15,15,15,15,15,3,0,0,0,0,0,0],
  \"wed\" : [0,0,0,0,0,0,0,15,15,15,15,15,15,15,15,15,15,15,15,3,0,0,0,0,0,0],
  \"thu\" : [0,0,0,0,0,0,0,15,15,15,15,15,15,15,15,15,15,15,15,3,0,0,0,0,0,0],
  \"fri\" : [0,0,0,0,0,0,0,15,15,15,15,15,15,15,15,15,15,15,15,3,0,0,0,0,0,0],
  \"sat\" : [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0],
  \"sun\" : [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0] }"}]
```



Example: Update Time/Date Group **td1** with new attributes

```
-u '{ "tmdategrp" : {
  "id":1,"name":"td1","description":"new description","disabled":0} }'
```



Example: Add new time/date to Time/Date Group **td1**

```
-u '{ "tmdate" : { "tmdategrpname":"td1","tmdate":{" \"range\" : {
  \"from\" : 1415851283, \"to\" : 1415887283 } } } }'
```



Example: To delete all times/dates from Time/Date Group **td1**

```
-d '{ "tmdate" : { "tmdategrpname":"td1" } }'
```



Example: To delete specified cmd from Time/Date Group **td1**

```
-d '{ "tmdate" : { "tmdategrpname":"td1", "tmdate" : "{ \"range\" : {
  \"from\" : 1415851283, \"to\" : 1415887283 } } } }'
```

Role Examples



Example: Retrieve list of Roles that match Role*

```
-g '{ "role" : { "name" : "Role*" } }'
  [{"id" : 0, "name" : "Role5", "rorder" : 3, "description" : "Desc3",
  "disabled" : 0, "risk" : 1, "action" : "A", "iolog" : "/tmp/iolog_XXXXXX",
  "script" : "accept;"}, {"id" : 1, "name" : "Role6", "rorder" : 2,
  "description" : "Desc3", "disabled" : 0, "risk" : 1, "action" : "A",
  "iolog" : "/tmp/iolog_XXXXXX", "script" : null}, {"id" : 2, "name" :
  "Role7", "rorder" : 1, "description" : "Desc3", "disabled" : 0, "risk" : 1,
  "action" : "A", "iolog" : "/tmp/iolog_XXXXXX", "script" : null}]
```



Example: Retrieve list of User Groups listed in the role Role6

```
-g '{ "roleusers" : { "name" : "Role6" } }'
  [{"id":1,"users":1,"type":"R"}, {"id":1,"users":1,"type":"S"}]
```



Example: Update role Role5 with new attributes

```
-u '{ "role" :
  {"id":0,"name":"Role5","rorder":3,"description":"Description
  4","disabled":0,"risk":1,"action":"A","iolog":"/tmp
  /iolog_XXXXXX","script":"accept;"},
  {"id":1,"name":"Role6","rorder":2,"description":"Desc3","disabled":0,"risk":1,"action":"A
  ","iolog":"/tm
  p/iolog_XXXXXX","script":null},
  {"id":2,"name":"Role7","rorder":1,"description":"Desc3","disabled":0,"risk":1,"action":"A
  ","iolog":"/tmp/io
  log_XXXXXX","script":null}'
```



Example: Add new Submit Host, *hostgrp2*, to role Role5

```
-u { "rolehosts" : { "name" : "Role5", "hostgrpname" : "hostgrp2" , "type"
  : "S" } }'
```



Example: To delete all User Groups from role Role5

```
-d '{ "roleusers" : { "name":"Role5" } }'
```



Example: To delete specified User Group from role **Role5**

```
-d '{ "roleusers" : { "name" : "Role5", "usergrpname": "ug1" } }'
```



Example: To delete all User Groups from role **Role5**

```
-d '{ "roleusers" : { "name": "Role5" } }'
```



Example: To delete specified User Group from role **Role5**

```
-d '{ "roleusers" : { "name" : "Role5", "usergrpname": "ug1" } }'
```



For more information on the **-b [-m <msg>]** setting in the *Privilege Management/etc/pb.settings* configuration file, please see *"Role Based Policy"* on page 52.

Client Registration Options

These options provide methods to create, maintain, export, and import role-based policies into Privilege Management for Unix and Linux, used with the `--rbp` option.

Usage

```
pbdbutil --reg [<options>] [ <file> <file> ...]
```

<code>-g { json param }</code>	Get Client Registration profile records.
<code>-u { json param }</code>	Update Client Registration profile records.
<code>-d { json param }</code>	Delete Client Registration profile records.
<code>-n</code>	Create new Client Registration profile database.
<code>-l</code>	List all Client Registration profiles in the database.

Client Registration Database Options

These options provide methods to define and maintain Client Registration profiles, used to simplify the registration of hosts within the Privilege Management enterprise.

```
"<-g|--get>" "{ param }"
```



Example: Retrieve Client Registration profile using the specified name:

```
-g '{ "name" : "default" }
    {"type":"settings","fname":"/etc/pb.settings"}
    {"type":"certificate","to":"/etc/${prefix}pbrest.pem${suffix}" }
    {"type":"save","sname":"networkencryption"}
    {"type":"save","sname":"restkeyencryption"}
    {"type":"save","sname":"sslservercertfile"}
    {"type":"save","sname":"sslserverkeyfile" }
```

```
"<-u|--update>" "{ param }"
```



Example: Update the specified Client Registration profile:

```
-u '{ "name" : "prof1", [{"type":"settings",
    "fname":"/etc/pb.mysettings"},
    {"type":"save","sname":"networkencryption"} ]' }
```

```
"<-d|--delete>" "{ param }"
```



Example: Delete the specified Client Registration profile:

```
-d '{ "name" : "profl" }'
```

```
<-l|--list>
```

List all the Client Registration profiles.

```
<-n|--new>
```

Create and initialize the Client Registration database.

Management Event Options

These options provide methods to create, maintain, export, and import role-based policies into Privilege Management for Unix and Linux.

Usage

- `pbdbutil --evt [<options>] [<file> <file> ...]`
- `-s { json param }`: Search Management event records
- **Record entities:**
 - `hostname`
 - `evtname`
 - `service`
 - `by`
 - `severity`
 - `before/after/then progname`
 - `version`
 - `arch`
 - `taxonomy`

Records can be searched using the above entities and are matched as wildcards.



Example:

```
-s '{ "taxonomy" : "chgmt" } 'Display all Change Management Events
```

```
-s '{ "taxonomy" : "chgmt", "hostname" : "host1" } ' Display all Change Management Events for host1
```

Description

`<-S|--searchvt> { json parameters }`

This option provides a method of retrieving change management events from the change management database.



Example: Retrieve all change management events

```
-S '{ "taxonomy" : "chgmt" }'
{"hostname" : "pbuild", "evtname" : "file_import", "service" : "pbdbutil9.0.0-01_debug",
"who" : "ctaylor", "severity" : 16, "progname" : "pbdbutil9.0.0-01_debug", "version" :
"9.0.0-01_debug", "arch" : "x86_64_linuxA", "data" : {"msg" : "foo, bar", "fname" :
"/opt/pbul/policies/pb.conf, conf", "version" : 4, "sid" : 4995, "pid" : 31976, "uid" :
```



```
0}, "utc" : "2014-11-1109 : 19 : 28"}
{"hostname" : "pbuild", "evtname" : "tag_file", "service" : "pbdbutil9.0.0- 01_debug",
"who" : "ctaylor", "severity" : 16, "programe" : "pbdbutil9.0.0- 01_debug", "version" :
"9.0.0-01_debug", "arch" : "x86_64_linuxA", "data" :{"fname" :
"/opt/pbul/policies/pb.conf", "tag" : "foo", "version" : -1, "sid" : 4995,"pid" : 31979,
"uid" : 0}, "utc" : "2014-11-11 09 : 19 : 30"}
```




Example: Retrieve change management events for *host1* only

```
-S '{ "taxonomy" : "chgmt" , "hostname" : "host1" }'
```


REST Keystore Options

Usage

```
pbdbutil --rest [<options>] [ <file> <file> ...]
```

-l	List all Application IDs in the database.
-d <appid>	Delete Application key.
-g <appid> [--svcname <name>] [<acl> ...]	<p>Create new Application key with ACLs Specify svcname to sync key across Service Group where acl is up to 8 regular expression strings in the form METHOD:/PATH/ATTRIBUTE where METHOD is GET, PUT, POST or DELETE.</p> <div style="border: 1px solid #ccc; padding: 10px; margin-top: 10px;"> <p> Example:</p> <pre>GET:/events</pre> <pre>PUT:/setting/</pre> <pre>POST:/key</pre> <pre>\(GET\ PUT\) :/setting/\(submitmaster\ acceptmasters\)</pre> <pre>-x <yyyy-mm-dd> Specify Application ID expiry</pre> </div>

Registry Name Service Options

These options allow the maintenance and interrogation of the Registry Name Services.

Usage

```
pbdbutil --svc [<options>] [ <file> <file> ...]
```

<code>-u '{ "svcgname" : "name", params... }'</code>	Create/Update Registry Name Service Group.
<code>-u '{ "cn" : "hname", params... }'</code>	Create/Update Registry Name Service Host.
<code>-u '{ "cn" : "hname", "uuid" : "", params... }'</code>	Create/Update external Host in Registry Name Service.
<code>-u '{ "svcgname" : "name", "cn" : "hname", params... }'</code>	Add/Update Registry Name Service Host to Service Group.
<code>-g '{ "svcgname" : "name" }'</code>	Retrieve Registry Name Service Group information.
<code>-g '{ "primary" : "name" }'</code>	Lookup the Primary Server within the Registry Name Service Group.
<code>-g '{ "cn" : "name" }'</code>	Retrieve Registry Name Service Host information by host common name.
<code>-g '{ "uuid" : "name" }'</code>	Retrieve Registry Name Service Host information by uuid.
<code>-d '{ "svcgname" : "name" }'</code>	Delete Registry Name Service Group.
<code>-d '{ "svcgname" : "name", "cn" : "name" }'</code>	Remove a host from Registry Name Service Group.
<code>-d '{ "cn" : "name" }'</code>	Delete Registry Name Service Host by host common name.
<code>-z <oldgrp> <newgrp></code>	Rename Registry Name Service Group.
<code>-l [<wildcard(s)>]</code>	List all the Registry Name Service Groups that match wildcard(s).
<code>-l</code>	Add an extra <code>-l</code> to list Servers in the Registry Name Service Group (s).
<code>-L [<wildcard(s)>]</code>	List all the Hosts that match wildcard(s).
<code>-L</code>	Add an extra <code>-L</code> to list Service Group membership and role.
<code>-p <svcgrp> <host></code>	Promote host to primary service within the specified Registry Name Service Group.
<code>-N [[<cn> [<port>]]</code>	Create and initialize Primary Registry Name Service database.
<code>-n</code>	Create new Registry Name Service database.

**Example: Recreate the Registry Name Service Database**

```
# pbdbutil --svc -N --force
```

**Example: List Service Groups from Registry Name Service Database**

```
# pbdbutil -P --svc -l
```

Result:

```
{
  "svcgid": 1,
  "svcgname": "registry_name_svc",
  "svc": "registry",
  "updated": "2016-06-09 15:42:33",
  "deleted": 0
}
{
  "svcgid": 2,
  "svcgname": "dfl_pb_policy_svc",
  "svc": "pbpolicy",
  "updated": "2016-06-09 15:42:33",
  "deleted": 0
}
{
  "svcgid": 3,
  "svcgname": "dfl_log_svc",
  "svc": "logsvr",
  "updated": "2016-06-09 15:42:33",
  "deleted": 0
}
{
  "svcgid": 4,
  "svcgname": "dfl_sudo_policy_svc",
  "svc": "sudopolicy",
  "updated": "2016-06-09 15:42:33",
  "deleted": 0
}
```

**Example: List primary and secondary servers within the Service Groups**

```
# pbdbutil -P --svc -l -l
```

Result:

```
{
  "svcgid": 1,
```



```

"svcgname": "registry_name_svc",
"svc": "registry",
"updated": "2016-06-14 10:43:14",
"deleted": 0,
"svcs": [
{
  "svcgid": 1,
  "hostid": 1,
  "role": "primary",
  "created": "2016-06-14 10:43:14",
  "updated": "2016-06-14 09:43:14",
  "deleted": 0,
  "cn": "pbuild",
  "uuid": "969ecab2-93d8-4322-a8cf-6314457053bb",
  "fqdn": "pbuild",
  "addrs": [
    {
      "family": 4,
      "port": 24351,
      "addr": "192.168.16.138"
    }
  ]
  "tnlzone": 0
}
]
}
{
  "svcgid": 2,
  "svcgname": "dfl_pb_policy_svc",
  "svc": "pbpolicy",
  "updated": "2016-06-14 10:43:14",
  "deleted": 0
}
{
  "svcgid": 3,
  "svcgname": "dfl_log_svc",
  "svc": "logsvr",
  "updated": "2016-06-14 10:43:14",
  "deleted": 0
}
{
  "svcgid": 4,
  "svcgname": "dfl_sudo_policy_svc",
  "svc": "sudopolicy",
  "updated": "2016-06-14 10:43:14",
  "deleted": 0
}
}
    
```



Example: Retrieve specified Service Group

```

pbdbutil --svc -g '{ "svcgname" : "registry_name_svc" }'
    
```



Result:

```
{"svcgid":1,"svcgname":"registry_name_svc","svc":"registry","updated":"2016-06-06  
16:56:53","deleted":0}
```



Example: Retrieve Specified Host by "cn"

```
pbdbutil --svc -g '{ "cn" : "pbuild" }'
```

Result:

```
{"addrs":  
[{"family":4,"addr":"192.168.16.138","port":24351}], "cn":"pbuild", "uuid":"969ecab2-93d8-  
4322-a8cf-6314457053bb", "$
```



Example: Retrieve Specified Host by "uuid"

```
pbdbutil --svc -g '{ "uuid" : "969ecab2-93d8-4322-a8cf-6314457053bb" }'
```

Result:

```
{"addrs":  
[{"family":4,"addr":"192.168.16.138","port":24351}], "cn":"pbuild", "uuid":"969ecab2-93d8-  
4322-a8cf-6314457053bb", "$
```



Example: Retrieve Host Entry for the Primary of the Specified Service Group

```
pbdbutil --svc -g '{ "primary" : "registry_name_svc" }'
```

Result:

```
{"svcgid":1,"svcgname":"registry_name_  
svc", "svc":"registry", "updated":1465228621, "deleted":0, "hostid":1, "role":"primary", "$
```



Example: Retrieve Host Information for Specified Host

```
pbdbutil --svc -L pbuild
```

Result:



```
{ "fqdn": "pbuild", "cn": "pbuild", "uuid": "969ecab2-93d8-4322-a8cf-6314457053bb", "adrs": [ {"addr": "192.168.16.138", "family": 4, $
```



Example: List All Hosts

```
# pbdbutil -P --svc -L
```

Result:

```
{
  "hostid": 1,
  "cn": "pbuild",
  "uuid": "969ecab2-93d8-4322-a8cf-6314457053bb",
  "fqdn": "pbuild",
  "adrs": [
    {
      "family": 4,
      "port": 24351,
      "addr": "192.168.16.138"
    }
  ],
  "tnlzone": 0,
  "updated": 1465897394,
  "deleted": 0
}
{
  "hostid": 4,
  "cn": "pbtest",
  "uuid": "969ecab2-93d8-4322-a8cf-6314457053bf",
  "fqdn": "pbtest",
  "adrs": [
    {
      "family": 4,
      "port": 24351,
      "addr": "192.168.16.184"
    }
  ],
  "tnlzone": 0,
  "updated": 1465898703,
  "deleted": 0
}
```



Example: Add a Specified Host ("cn" Common Name and "uuid" Are Required)

```
# pbdbutil --svc -u '{ "cn" : "pbtest" , "uuid" : "969ecab2-93d8-4322-a8cf-6314457053bf" }'
```

**Example: Add New Service Group**

```
# pbdbutil --svc -u '{ "svcgname" : "foobar", "svc" : "logsvr" }'
```

**Example: Add Host to Service Group**

```
# pbdbutil --svc -u '{ "svcgname" : "foobar", "cn" : "pbtest" }'
```

**Example: Delete Host**

```
# pbdbutil --svc -d '{ "cn" : "pbtest" }'
```

**Example: Add Host to Service Group as Primary Server**

```
# pbdbutil --svc -u '{ "svcgname" : "foobar", "cn" : "pbtest", "role" : "primary" }'
```

**Example: Delete Host When It Is a Primary**

```
# pbdbutil --svc -d '{ "cn" : "pbtest" }'
```

4011.01 Host is a primary server. Please reassign primary before deleting host from service group, or use *force* on the \$.

**Example: Delete the Service Group**

```
# pbdbutil --svc -d '{ "svcgname" : "foobar" }' --force
```

**Example: Promote a Host That Is Currently a Secondary Server to a Primary Server**

```
# pbdbutil --svc -p foobar pbtest
```

Database Synchronization Options

These options allow the interrogation of Database Synchronization status on primary servers.

Usage

```
pbdbutil --dbsync [<options>] [ <file> <file> ...]
```

-l	List Database Synchronization history.
-l [<dbfile(s)>]	List outstanding Database Synchronization entries.
-c <dbfile(s)>	Deletes the _pblog entries from the specified database. Each database that can be synchronized (FIM, Sudo, RBP, etc) has a _pblog table used to synchronize data from the primary to the secondaries. When adding a FIM configuration, or a new role, or a pbsudo host, then the corresponding INSERT or UPDATE SQL is added in the _pblog table.
-R <svc> [<cn>]	Initiates a synchronize on database immediately even if there is no change, for specified service.
-A <svcname> <...>	Set databases in Service Group(s) as being automatically synchronized.
-X <svcname> <...>	Unset databases in Service Group(s) as being automatically synchronized.

Registry Name Service Cache Options

Each host has a Registry Name Service Cache that holds the Service Group information that is applicable to them.

These options allow the retrieval of information and options to re-initialize the Registry Name Service Cache database.

Usage

```
pbdbutil --scache [<options>] [ <file> <file> ...]
```

--cn	Retrieve Common Name from the Registry Name Service.
-w	Retrieve my Registry Name Service information.
-l	List all the locally cached Registry Name Service entries.
-s <[- +]attribute>	Sort the list of records by attribute name (asc/desc).
-R	Refresh the local Registry Name Service cache.
-N { param }	<p>Create and initialize the Primary Registry Name cache database where the { param } argument is formatted JSON with parameters:</p> <ul style="list-style-type: none"> • "hostname" : "host1": hostname of the Registry Manager REST service • "port" : 24351: port of the Registry Manager REST service • "appid" : "appid": appid of the Registry Manager REST service • "appkey" : "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx": appkey of the Registry Manager REST service
-m <msg>	Specify message. Required when change management enabled.

File Integrity Monitor Options

These options provide maintenance for the File Integrity Monitor database, and options for the client to run an integrity check.

Usage

```
pbdbutil --fim [<options>] [ <file> <file> ...]
```

Options for FIM Client

-r	Run FIM check.
-U	Run FIM check and update database.

Options for FIM Server Database Management

-n	Create new FIM database.
-l	List all FIM configurations in database.
-l	Add an extra -l to list host assignments.
-s <[- +]attribute>	Sort the list of records by attribute name (asc/desc).
-i <file>	Import FIM configuration file.
-e <name> <file>	Export specified FIM configuration.
-g <name>	Get FIM configuration by name.
-d <name>	Delete FIM configuration.
-d { "cfg" : { "name" : "<wildcard>" } }	Delete FIM configuration matching wildcard.
-u { "name" : "<name>", "cfg": { json param... } }	Update FIM configuration.
-A <name> <host(s)>	Assign host to configuration.
-X <host(s)>	Unassign host from configuration.
-g { "rpt" : { "uuid" : "<uuid>" } }	Get specified FIM report.
-g { "rpt" : { params ... } }	Retrieve report summarized from multiple reports. See below for attributes.
--format '["header", "header2", ...]'	Define retrieved fields when using CSV report.
-d { "rpt" : { "uuid" : "<uuid>" } }	Delete FIM report.
-d { "rpt" : { params ... } }	Delete FIM report(s) - see below for attributes.

```
-L [{ Retrieve, List or Delete FIM reports, with attributes:  
["name" : "<wildcard>"],  
["uuid" : "<uuid>"],  
["host" : "<wildcard>"],  
["older" : <epoch>],[ "newer" : <epoch>],  
["older" : { "years" : n, "months" : n, "days" : n, "hours" : n }  
["newer" : { "years" : n, "months" : n, "days" : n, "hours" : n  
]  
["updates" : <bool>],  
["risk" : <lvl>],  
["risk_higher" : <lvl>],  
["risk_lower" : <lvl>],  
["regex" : true}]
```

```
-s <[-|+]attribute>
```

Sort the list of records by attribute name (asc/desc).

Event Logfile Cache Options

These options allow users to query and maintain the database that caches the event logfile names for use with BeyondInsight for Unix & Linux.

Usage

```
pbdbutil --evtcache [<options>]
```

Display cached list of event log files:

<code>--lstcache={param...}</code>	Use equals sign (=) for optional parameter.
<code> -S[{param...}]</code>	No space between switch name and optional parameter.
<code>-s <[- +]attribute></code>	Sort the list of records by attribute name (asc/desc).

Where the `{ param }` argument is formatted JSON parameters:

- `{"path": "<pattern>"}`: glob wildcard for logfile path
- `{"runhost": "<host1>"}`: Filter by runhost name
- `{"loghost": "<host2>"}`: Filter by loghost name
- `{"from": "<yyyy-mm-dd HH:MM>"}`: Filter by event logs active on or after this date/time
- `{"to": "<yyyy-mm-dd HH:MM>"}`: Filter event logs active on or before this date/time
- `{"start": <offset>`: Specify record offset (number) when limiting output
- `{"len": <limit>`: Specify number of rows when limiting output

Remove event log file entries from the logfile cache database:

```
-d '{param...}'
```

Where the `{ param... }` argument is formatted JSON parameters:

- `{"path": "pattern"}`: glob wildcard for logfile path
- `{"loghost": "host2"}`: Filter by loghost name
- `{"from": "<yyyy-mm-dd HH:MM>"}`: Filter event logs active on or after this date/time
- `{"to": "<yyyy-mm-dd HH:MM>"}`: Filter event logs active on or before this date/time

I/O Logfile Cache Options

These options allow users to maintain the database that caches the I/O logfile names for use with BeyondInsight for Unix & Linux.

Usage

```
pbdbutil --iocache [<options>]
```

Display cached list of I/O log files:

<code>--lstcache=[{ param... }]</code>	Use equals symbol (=) for optional parameter.
<code> -S[{ param... }]</code>	No space between switch name and optional parameter.
<code>-s <[- +]attribute></code>	Sort the list of records by attribute name (asc/desc).

Remove I/O log file entries from the logfile cache database:

```
-d '{param...}' ]
```

Where the `{ param... }` argument is formatted JSON parameters:

- `{"path":"pattern"}`: glob wildcard for logfile path
- `{"loghost":"host1"}`: Filter by loghost name
- `{"submithost":"host2"}`: Filter by submithost name
- `{"runhost":"host3"}`: Filter by runhost name
- `{"submituser":"user1"}`: Filter by submituser name
- `{"runuser":"user2"}`: Filter by runuser name
- `{"runcmd":"command"}`: Filter by run command
- `{"from":"<yyyy-mm-dd HH:MM>"}`: Filter I/O logs created on or after this date/time
- `{"to":"<yyyy-mm-dd HH:MM>"}`: Filter I/O logs created on or before this date/time
- `{"start":"<offset>"}`: Specify record offset (number) when limiting output
- `{"len":"<len>"}`: Specify number of rows when limiting output

Migrate I/O log location cache database (Upgrades only):

```
-n [--force]
```

Migrates pre-v10.3.1 I/O log location cache database to the new database configuration.

Migration is automatically done during an upgrade via `pbinstall`. Running this manually is typically not necessary unless circumstances prevented the automatic migration during the upgrade. The optional `--force` skips backup of the original/obsolete I/O log cache database if it already has been backed up by other methods.

Where the `{ param }` argument is formatted JSON with parameters:

- `"path":"pattern"`: glob wildcard for logfile path
- `"loghost":"host2"`: loghost name

- **"submithost":"host1"**: Filter by submithost name
- **"runhost":"host1"**: Filter by runhost name
- **"submituser":"user1"**: Filter by submituser name
- **"runuser":"user1"**: Filter by submituser name
- **"runcmd":"cmd"**: Filter by runcmd name
- **"from":"<yyyy-mm-dd HH:MM>"**: Filter by logfiles opened on or after this date/time
- **"to":"<yyyy-mm-dd> HH:MM"**: Filter by logfiles opened on or before this date/time
- **"start":<offset>**: Specify record offset (number) when limiting output
- **"len":<limit>**: Specify number of rows when limiting output

I/O Logfile Queue Options

These options allow users to maintain the database that queues the I/O logfile names for indexing to Solr.

Usage

```
pbdbutil --iologidx[<options>]
```

-l	List queued iolog files.
-d <wildcard spec>	Delete queued iolog files.

When an **iolog** is started, **pblogd** (or **pbmaterd**) adds that **iolog** to the logfile queue with a **pblogd_status** of **started** and **retry** set to **never**. When the **iolog** file is closed in a normal fashion, the **pblogd_status** is set to **finished**. During the time that **pblogd** is active, it periodically sends a heartbeat. When an **iolog** is not properly closed (**pbrun** killed or network issues, for example), the heartbeat is used in conjunction with the **iologactionqueuetimelimit** keyword to artificially set the **pblogd_status** to **finished**, so that **iolog** can be processed for Solr or **iologcloseaction**. When an **iolog** is being processed for Solr or **iologcloseaction**, the **proc_status** is set to **processing**. When Solr or **iologcloseaction** has successfully completed, the **proc_status** is set to **finished**. When Solr reports a recoverable error, or **iologcloseaction** returns **-1**, the **iolog** is re-queued by setting **lastupdated** to **now**, setting **retry** (**now** + **iologactionretry** minutes), and incrementing the **retries**.



Example: Delete all queued I/O log file names

```
pbdbutil --iologidx -d \*
```

Integrated Product Options

These options provide options to configure the Integrated Products Queue database.

Usage

```
pbdbutil --intprod [<options>] [ <file> <file> ...]
```

-l	List all entries in Integrated Product database queue.
-d <wildcard>	Delete entries from Integrated Product database queue.

poldbg

Description

Policy language debugging can be enabled, disabled, and reviewed using the **poldbg** option. With this command, you can list policy debugging entries in an attempt to identify and resolve issues that may have occurred within a policy. In addition, you can specify users whose policy is debugged, and can even specify the amount of time that debugging is enabled for that user and policy. The user executes the **pbrun** command, and the administrator can then review the debugging information.



Note: Policy debugging is only available for if statements and switch case statements.

Syntax

Run to list debugging policy entries.

```
--poldbg -l
```

Run to identify users who can debug entries. You can also designate how long the user has access.

```
--poldbg -u
```

Run to view and print a clean output of events for policy debugging in JSON-equivalent format.

```
pbadmin -P --evt -s '{taxonomy" : policydbg" }'
```

Run to view and print events for policy debugging in a CSV-type format.

```
pbadmin -C --evt -s '{taxonomy" : policydbg" }'
```



Example:

```
pbadmin --poldbg -u rjones 2h
```

In this example, the user **rjones** is specifically allowed to debugging access for two hours.



Example:

```
pbadmin -C --evt -s '{taxonomy" : policydbg", "rowid" : 3 }'
```

In this example, the events are going to be provided in a CSV-type format in which the information specifically in row three is expanded.

Elasticsearch Credential Management

The pbdbutil settings outlined here support Elasticsearch credential management.

The options available with the pbdbutil tool are also available in the PMUL REST API.

i For more information, please see "[Elasticsearch Logstash API Calls](#)" on page 565.

Usage

```
pbdbutil --elkcred [<options>]
```

<code>--elkcred -g <id></code>	<p>Retrieves a credential by ID. The credential is output in JSON format. Use -P to make the output more readable.</p> <p>The REST API call: elkcred -X GET</p>
<code>-s '{ "id": "<id>", ... }'</code>	<p>Adds a credential. The response is <i>OK</i> if the credential is set successfully; otherwise, a relevant error message is displayed.</p> <p>The REST API call: elkcred -X PUT</p>
<code>-d <id></code>	<p>Deletes a credential. The response is <i>OK</i> if the credential is deleted successfully; otherwise, a relevant error message is displayed.</p> <p>The REST API call: elkcred -X DELETE</p>
<code>-l</code>	<p>Lists all credentials. The credential is output in JSON format. Use -P for more readable output.</p> <p>The REST API call: elkcreds -X GET</p>
<code>-t <id></code>	<p>Tests an existing credential. Test results are shown in JSON format. Use -P for more readable output.</p> <p>Tests of a token or apikey credential fail against Logstash instances.</p> <p>The REST API call: elkcredtest -X GET</p>
<code>-t '{ "id": "<id>", ... }'</code>	<p>Tests a prospective credential with the values currently in <code>/etc/pb.settings</code> (e.g., <code>elkinstances</code>), use the JSON fields relevant to the credential type.</p> <p>i For more information on credential types, please see "Credential Store" on page 351.</p> <p>The REST API call: elkcredtest -X POST.</p> <p>To test a credential independently of <code>/etc/pb.settings</code>, add the <code>elkinstances</code> JSON attribute. See "Test a Credential" on page 419 for the example code snippet.</p> <p>The REST API call: elkcredtest -X POST</p>

Test a Credential

To test a credential independently of `/etc/pb.settings`, add the `elkinstances` JSON attribute, as shown here:



Example:

```
# pbdbutil --elkcred -t '{"id": "elastic_token", "type": "token", "username": "jeff", \
"password": "<password>", "endpoint": "/_security/oauth2/token", \
"elkinstances": "elasticsearch=https://elksite.us-east-1.aws.found.io}"' -P
{
  "results": [
    {
      "token-request": {
        "url": "https://elksite.us-east-1.aws.found.io/_security/oauth2/token",
        "curlcode": "0 (No error)",
        "httpcode": "200 (OK)"
      },
      "test-request": {
        "url": "https://elksite.us-east-1.aws.found.io/?pretty",
        "curlcode": "0 (No error)",
        "httpcode": "200 (OK)"
      }
    }
  ]
}
```

The format of the `elkinstances` value is the same as it would be in `/etc/pb.settings`.

Use `-P` to make the output more readable. As is the case with an existing credential, an attempt to test a prospective credential of type `token` or `apikey` against a Logstash instance fails.

pbencode

pbencode reads a file and encrypts it using the encryption key that is specified on the command line or in the settings file. The encrypted result is displayed on the standard output.



IMPORTANT!

pbencode should be used only on files that are backed up. There is no way to decrypt a file after it has been encrypted.

Syntax

- **Version 3.5 and earlier:** long command options not supported.
- **Version 4.0 and later:** long command options supported.

```
pbencode [options]
  -f, --file=file_name
  -E, --encryptiontype=policy|report|settings
pbencode -v | --version
pbencode -h | --help
```

Arguments

-f, --file=file_name	The input file to encode. The default is the policy configuration file.
-E, --encryptiontype =policy report settings	The encryption type to use, based on the definition set in the settings file.
-v, --version	Optional. Print the pbencode version and exit.
-h, --help	Optional. Display the pbencode help message and exit.

Files

The files used may appear on the command line.



Example:

```
pbencode -f /opt/pbul/policies/pb.conf -E policy > /etc/pb.conf.enc
```



Encodes the policy file with the encryption type and key that is specified in the settings file and places the result in `/etc/pb.conf.enc`.



For more information, please see "`pbkey`" on page 426.

pbguid

The **pbguid** command runs the server for the browser-based Privilege Management for Unix and Linux GUI. This program enables you to change settings files, view event and I/O logs (including archived I/O logs), edit policy configuration files, run the event log report writer, and update the GUI configuration. a socket-listener process (typically **inetd**, **xinetd**, or **pbguid -d -p**) starts **pbguid**.

changes that are made to the **pb.settings** file after the daemon is started will not affect the operation of the daemon. If you change the **pb.settings** file, then you must restart the daemon for the changes to take effect. If you do not restart the daemon, then the daemon continues to operate using a snapshot of the **pb.settings** file that was cached at the time the daemon was started.

Start the GUI by pointing a browser at a machine with the GUI installed and using the appropriate protocol. For example, on a machine called **pbguidhost** with the GUI running on port 24348 using HTTP, a user would type <http://pbguidhost:24348> in the browser.

The GUI user is authenticated by checking the entered user name and password against the Unix/Linux passwords on the host that is running **pbguid**.

The user is authorized by a request that is sent from **pbguid** to the policy server **pbmasterd** with the **pbclientname** of **pbguid** and the variables **browserip** and **browserhostname** set. The browser variables might reflect the proxy name if the browser uses a proxy to get to the **pbguid** machine.

The type of requested activity is indicated by arguments that are sent to **pbmasterd**. These arguments are checked against the policy file to determine if the user is allowed to conduct the activity.

For all activities:

- The value of the **pbclientname** variable is set to **pbguid**.
- The value of the command variable is set to **pbguid**.
- The value of **argv[0]** is set to **pbguid** (or the prefix/suffix version of the **pbguid** command name). Each activity is identified by the values of **argv[1]**, **argv[2]**, and **argv[3]**, as shown in the following table.

activity	argv[1]	argv[2]	argv[3]
Selecting a file with File browser	browse		
Viewing an I/O log	iolog	file name of I/O log	
Viewing an event log	log		
Updating the settings file	settings		
Updating the GUI configuration options	defaults		
accessing the Policy Editor	policy		
Saving a policy file	save	policy file name	
Saving a policy file to a different file name	report	new policy file name	original policy file name
accessing the event log reporting pages	report		
creating a new event log report set	report	edit	
Selecting event log fields for a report set	report	select	
Editing an existing event log report set	report	edit	report set file name
Executing an event log report set	report	exec	report set file name
Retrieving information about an event log report set	report	info	report set file name
Saving an event log report set	report	save	report set file name
accessing the entitlement reporting pages	entitlement		

activity	argv[1]	argv[2]	argv[3]
creating a new entitlement report set	entitlement	edit	
Editing an existing entitlement report set	entitlement	edit	report set file name
Executing an entitlement report set	entitlement	exec	report set file name
Retrieving information about an entitlement report set	entitlement	info	report set file name
Saving an entitlement report set	entitlement	save	report set file name
access the Task Manager	console		


You can distinguish the *accessing the event log reporting pages* activity from the other event log reporting activities by checking the value of **argc**, which is **2** for the accessing activity and **3** or **4** for the other activities. Use a similar technique to distinguish the *accessing the entitlement reporting pages* activity from the other entitlement reporting activities.

Syntax

- **Version 3.5 and earlier:** long command options not supported.
- **Version 4.0 and later:** long command options supported.

```
pbguid [options]
  -d, --daemon
  -e, --error_log=log_file
  -f, --foreground
  -p, --port=port_number
  -S, --secure
pbguid -v | --version
pbguid -help
```

arguments

-e, --error_log=log_file	Optional. Specifies a diagnostic log.
-S, --secure	Optional. Uses HTTPS rather than HTTP as the protocol. This option provides a more secure mode of operation. <div style="border: 1px solid black; padding: 5px; margin-top: 10px;">  Note: In version 6.0 and later, the --https long command option is no longer valid. Use --secure instead. </div>
-d, --daemon	Optional. Run as a stand-alone daemon instead of from inetd or xinetd . This option can be specified only with the port number. For example: <pre style="background-color: #f0f0f0; padding: 5px; margin-top: 10px;">pbguid -d -p</pre>
-f, --foreground	Run as a foreground job instead of forking and dissociating from the current job. This is the most useful for running pbguid from inittab .
-p, --port=port_number	Optional. Listens to the specified port, instead of the default, when running as a stand-alone daemon. This option requires -d .

-v, --version	Optional. displays the pbguid version and exits.
--help	displays the pbguid help message and exits.

Files

- Privilege Management for Unix and Linux settings file.
- Privilege Management for Unix and Linux configuration file.

i For more information, please see the following:

- "[pbbench](#)" on page 284
- "[pbcheck](#)" on page 371
- "[pbkey](#)" on page 426
- "[pbmasterd](#)" on page 441
- "[pbreplay](#)" on page 454
- On the **pbclientname**, **command**, **argv**, and **argc** variables, the [Privilege Management for Unix and Linux Policy Language Guide](https://www.beyondtrust.com/docs/privilege-management/unix-linux/index.htm) at <https://www.beyondtrust.com/docs/privilege-management/unix-linux/index.htm>

pbhostid



*Note: **pbhostid** has been deprecated in version 10.0. Use **pbadmin --info --uuid**.*

pbkey

The **pbkey** program generates an encryption key that is suitable for any of the Privilege Management encryption algorithms and stores it in a file that is specified on the command line or in the settings file. If **pbrun**, **pbmasterd**, or **pblocald** find the file **/etc/pb.key**, then they use it to encrypt data that is sent to the other programs.

If encryption is used, then the Privilege Management for Unix and Linux programs use the key that is specified in the settings file to encrypt local data and network traffic.

For network traffic, the contents of this file must be the same on all machines that are running Privilege Management for Unix and Linux for requests to execute. The file should be owned by root and have permissions set so that only root can read or write the file.

Syntax

- **Version 3.5 and earlier:** long command options not supported.
- **Version 4.0 and later:** long command options supported.
- **Version 8.5 and later:** **-F** option added.

```
pbkey [options]
    -f, --keyfile=key_file_name
    -F, --seckeyfile=key_file_name
pbkey -v | --version
pbkey -h | --help
```

Arguments

f, --keyfile=key_file_name	Optional. The name of the key file to create. The default is the value that is specified in the settings file or /etc/pb.key .
-F --seckeyfile=key_file_name	Optional. The name of the new high security key file to create. This -F --seckeyfile=key_file_name key file format must be used when enhanced security is required. Available in v8.5 or later.
-v, --version	Optional. Displays the pbkey version and exits.
-h, --help	Optional. Displays the program's help message and exits.

Files

key_file_name	The resulting key file.
----------------------	-------------------------



Example: Executing the command generates a new key and puts it into the file **/etc/pb.key**:

```
pbkey /etc/pb.key
```



For more information, please see the following:

- *"pbcheck" on page 371*
- *"pbhostid" on page 425*
- *"pblocald" on page 430*
- *"pbmasterd" on page 441*
- *"pbpasswd" on page 446*
- *"pbreplay" on page 454*
- *"pbrun" on page 465*
- *"pbsum" on page 471*

pbless

Version 4.0.0 and later: **pbless** setting available.

The **pbless** pager is similar to the **less** pager. It has been modified so that it can be used securely with the Privilege Management for Unix and Linux programs. Security is enhanced by the following features:

- **pbless** must be started with a full path name specified for the file to be read.
- The user cannot access any files other than the one that is specified at startup time.
- The user is not allowed to spawn any processes.

This program, when used in conjunction with Privilege Management for Unix and Linux, allows users to access a specific file as root but not to access other **root** functions.

Syntax

```
pbless fullpathname
```

Arguments

```
fullpathname
```

File to display

Files

None



Example: Display the contents of the file called **filename**:

```
pbless filename
```



For more information, please see "[pbrun](#)" on page 465.

pblicense



Note: *pblicense* has been deprecated in version 10.0 of Privilege Management for Unix and Linux. Please use *pbadmin* with the *--lic* option to maintain licenses.

pblocald

pblocald is the local daemon that runs programs, when instructed, by the appropriate policy server daemon. A socket-listener process (typically **inetd**, **xinetd**, Solaris SMF, or **pblocald -d**) starts **pblocald**. **pblocald** checks the command line arguments (**-m** or **--accept_masters**), the **acceptmasters** setting in the settings file, or the netgroup **pbacceptmasters** to determine the policy server hosts from which it accepts requests.

Requests from policy server daemons that are not in this list are refused. **pblocald** logs all diagnostic messages in the log file that is specified by the **-e** command line argument or by the **pblocaldlog** setting.

Changes that are made to the **pb.settings** file after the **pblocald** daemon is started will not affect the operation of the daemon. If you change the **pb.settings** file, then you must restart the daemon for the changes to take effect. If you do not restart the daemon, then the daemon continues to operate using a snapshot of the **pb.settings** file that was cached at the time the daemon was started.


Syntax

- **Version 3.5 and earlier:** long command options not supported.
- **Version 4.0 and later:** long command options supported.

```
pblocald [options]
-a, --syslog_accepts
-m, --accept_masters=host_list
-d, --daemon
-D, --debug=<level>
-e, --error_log=log_file_name
-f, --foreground
-i, --info <argument placeholder characters>
-p, --port=port_number
-s, --syslog
-V, --check_version
pblocald -v | --version
pblocald --help
```

Arguments

-a, --syslog_accepts	Optional. Records accepted tasks in the syslog. Version 6.2 and earlier: option not available. Version 7.0 and later: option available.
-m, --accept_masters=host_list	Optional. A list of policy server hosts from which pblocald accepts secured task requests. The list can include hostnames, IP addresses, DNS SRV lookups, and external program specifications
-d, --daemon	Optional. Runs as a standalone daemon instead of from inetd or xinetd . This mode listens to the port that is defined by the -p command line argument or in the localport setting.
-D, --debug=<level>	Generate debug trace logs in the same directory pointed to by pblocaldlog . Version 7.5 and earlier: option not available. Version 8.0 and later: option available.

-e, --error_log=log_file_name	Optional. Records diagnostic messages in the file logfile instead of using the settings file entry pblocald-log .
-f, --foreground	pblocald normally spawns a child process and dissociates from the job that it starts. Although this method is beneficial when running from inetd , xinetd , or the command line, it stops pblocald from running under the init daemon (from /etc/inittab). This option prevents pblocald from dissociating and allows it to run from the inittab .
-i -i-info <argument placeholder characters>	<p>On Linux and AIX, the pblocald process replaces the argument placeholder characters with the following information about the submitting request:</p> <ul style="list-style-type: none"> • submitting user • submit host • pbrun's pid • runuser • runargv <p>The format is:</p> <pre>submituser@submithost pid runuser: runargv</pre> <p>This allows an administrator to use the ps command to view more information about the running pblocald processes.</p> <div style="border: 1px solid black; background-color: #e6f2ff; padding: 5px; margin-top: 10px;">  Note: This feature is not available on HP-UX and Solaris. </div>
-p, --port=port_number	Optional. When running as a standalone daemon, listens to the provided port instead of the default.
-s, --syslog	<p>Optional. Sends error messages to syslog in addition to the diagnostic message file.</p> <p>-s works only if /etc/syslog.conf is configured to have syslog process auth.err (or less severe) messages.</p>
-V, --check_version	Optional. Records diagnostic messages if a connecting client version does not match the pbmasterd version
-v, --version	Optional. Displays the program version and exit.
--help	Optional. Displays the program's help message and exit.

Files

The **/etc/pb.settings** file that contains a list of valid **acceptmasters** hosts.

i For more information, please see the following:

- "[pbcheck](#)" on page 371
- "[pbhostid](#)" on page 425
- "[pbkey](#)" on page 426

i

- *"pbmasterd" on page 441*
- *"pbpasswd" on page 446*
- *"pbreplay" on page 454*
- *"pbrun" on page 465*
- *"pbsum" on page 471*

pblog

The **pblog** program selectively displays entries from an event log. Each time a job is accepted, rejected, or completed, or a keystroke action event occurs, an entry is appended to the event log file. The event log file is specified by the “**authvt=**” label in **eventdestinations** setting or the **eventlog** setting in the settings file, or by the **eventlog** variable in the Privilege Management for Unix and Linux policy file. By default, the **eventlog** variable is set to **/var/log/pb.eventlog**, **/usr/log/pb.eventlog**, **/var/adm/pb.eventlog**, or **/usr/adm/pb.eventlog**, depending on the operating system.

With no command line arguments, **pblog** reads and displays all entries in the default event log file. You can specify a different event log with the **-f** or **--eventlog** argument. You can specify a decryption key file with the **-k** or **--keyfile** argument.

Syntax

- **Version 3.5 and earlier:** long command options not supported.
- **Version 4.0 and later:** long command options supported.

```
pblog [options]
  -a, --accept_format=expression
  -c, --constraint=expression
  -d, --dump
  -e, --finish_format=expression
  -f, --eventlog=file_name
  --db
  --ff
  --odbc
  -i, --keystroke_format=expression
  -k, --keyfilefile_name
  -l, --verbose
  -o, --dbout=file_name>
  -O, --odbcout=<dsn>
  -D, --diff
  -p, --all_formats=expression
  -q, --quiet
  -r, --reject_format=expression
  -t, --tail
pblog -X|--xml [options]
  -c, --constraint=<expression>
  -F, --field_list=field_names
  -f, --eventlog=file_name
  -k, --keyfile=file_name
pblog -C|--csv[options]
  -c, --constraint=expression
  -F, --field_list=field_names
  -f, --eventlog=file_name
  -H, --csv_header
  -k, --keyfile=file_name
  -S, --csv_separator
pblog -J, --json
  -P, --pretty
pblog -v|--version
pblog --help
```

Arguments

-a, --accept_format=expression	Optional. Format expression for Accept events.
-c, --constraint=expression	Optional if the expression is the last argument on the command line. Constrains entries according to the expression.
-C, --CSV	Optional. Produces comma-separated value (CSV) output.
-d, --dump	Optional. Dumps entries as they are read.
-e, --finish_format=expression	Optional. Format expression for finish events.
-f, --eventlog=file_name	Optional. Alternate event log to report.
--db	Optional. Alternate event log is a database.
--ff	Optional. Alternate event log is a flat file.
--odbc	Optional. Alternate event log is a ODBC DSN.
-F, --field_list=field_names	Optional field list to be displayed for CSV and XML output. When specifying multiple field names, enter them as comma-separated values (no spaces).
-H, --csv_header	Optional. Produces a field-name header in CSV mode.
-i, --keystroke_format=expression	Optional. Format expression for keystroke action events.
-k, --keyfile=file_name	Optional. Name of the encryption key file. The file_name specified needs to be listed as one of the keyfile in eventlogencryption keyword.
-l, --verbose	Optional. Turns on verbose mode. Prints all variables, one per line.
-o, --dbout=file name>	Optional. Copy missing records to database.
-O, --odbcout=<dsn>	Optional. Copy missing records to ODBC.
-D, --diff	Optional. Report differences instead of copying.
-p, --all_formats=expression	Optional. Print format expression for all events.
-q, --quiet	Optional. Turns on quiet mode. Do not report expression errors (for example, undefined variables).
-r, --reject_format=expression	Optional. Format expression for Reject events.
-S, --csv_separator	Optional. Field separator for CSV mode (default is comma).
-t, --tail	Optional. Tail mode. Loop and report new entries as they occur (forces dump mode).
-J --json	Optional. Produces output in JSON format.
-P --pretty	Optional. Pretty print.

-v, --version	Optional. Displays the pblog version and exits.
-X, --xml	Optional. Produces XML-formatted output.
--help	Optional. Displays the program help message and exit.

i For more information about syntax to specify multiple encryption algorithms and files, please see "[eventlogencryption](#)" on page 202.

Default Output Expressions

Accept Dump	<pre>printf('%s %s %s %s %s@%s -> %s@%s\n\t%s', uniqueid, event, date, time, user, submithost, runuser, runhost, join(runargv))</pre>
Reject Dump	<pre>printf('%s %s %s %s %s@%s\n\t%s', uniqueid, event, date, time, user, submithost, join(argv))</pre>
End/Finish Dump	<pre>printf('%s %s %s', uniqueid, event, exitstatus)</pre>
Accept	<pre>printf('%s %s %s %s@%s -> %s@%s\n\t%s\n\t%s', event, date, time, user, submithost, runuser, runhost, join(runargv), exitstatus)</pre>
Reject	<pre>printf('%s %s %s %s@%s\n\t%s', event, date, time, user, submithost, join(argv))</pre>
End/Finish	<pre>printf('%s %s %s', uniqueid, event, exitstatus)</pre>
Keystroke	<pre>printf('%s %s %s %s %s', event, keystrokestatus, keystroke, keystroke, keystroke)</pre>

Read an Event Log



Example: If `pb.settings` file has:

```
#eventdestinations
eventlog /var/log/pb.eventlog
```

pblog with no arguments reads the flat file event log specified in the `eventlog` setting:



```
# pblog
```



Example: If `pb.settings` file has:

```
eventdestinations    authevt=db
eventlog              /var/log/pb.eventlog.db
```

pblog with no arguments reads the SQLite DB event log specified in the `eventlog` setting:

```
# pblog
```



Example: Read a SQLite DB event log whose path is specified in the event log setting:

```
# pblog
```



Example: Read a specific SQLite DB event log:

```
# pblog -f /path/to/mypb.eventlog.db --db
```



Example: Read a specific flat file event log

```
# pblog -f /path/to/mypb.eventlog.flat --ff
```



Example: Read an ODBC type event log:

```
# pblog -f MyDSN --odbc
```

In this case, `odbc.ini` and `odbcinidir` files located in the directory specified in `odbcinidir` setting (default `/etc/pbul/etc`) will be read to get the connection information to the MySQL or Oracle database.



For more information, please see the default values listed in "eventlog" on page 118.

Copy Event Log Records

If multiple targets for authorized events are defined in **eventdestinations**, or if you want to copy event log records from one event log file to another, the options **-o** (output to SQLite database) or **-O** (output to ODBC database) can be used. This only copies the event log records with a uniqueid that does not exist in the destination.



Example: Copy from a flat file to a SQLite database:

```
# pblog --ff -f /var/log/pb.eventlog.flat -o /var/log/pb.eventlog.db
```



Example: Copy records from a flat file to the ODBC database:

```
# pblog --ff -f /var/log/pb.eventlog.flat -O MyDSN
```

Where **MyDSN** is the ODBC Data Source Name whose connection information to the MySQL or Oracle database is configured in the **odbc.ini/odbcinst.ini** files (see **odbcinidir** setting).



Example: Copy records from a SQLite database to the ODBC database:

```
# pblog --db -f /var/log/pb.eventlog.db -O MyDSN
```

Where **MyDSN** is the ODBC Data Source Name whose connection information to the MySQL or Oracle database is configured in the **odbc.ini/odbcinst.ini** files.



For more information, please see "**odbcinidir**" on page 121.

Report Difference Between Event Log Destinations:

If multiple event destinations were used, and you want to report on records that might be in one destination but not another, you can use **-o**, **-O** with **-D** option:



Example: Report differences between event log records in a flat file versus a SQLite database:

```
# pblog --ff -f /var/log/pb.eventlog.flat -o /var/log/pb.eventlog.db -D
uniqueid,etype,epoch
ac1420215df2ac3604C5,Reject,2020/02/20 13:08:06
ac1420215df2ac3604C7,Accept,2020/02/20 13:08:06
ac1420215df2ac3604C7,Finish,2020/02/20 13:08:54
ac1420215df2ac3704C9,Accept,2020/02/20 13:08:07
ac1420215df2ac3704C9,Finish,2020/02/20 13:08:55
```



Example: Report differences between event log records in a flat file versus a MySQL database:

```
# pblog --ff -f /var/log/pb.eventlog.flat -O MyDSN -D
uniqueid, etype, epoch
ac1420215df2ac3604C5, Reject, 2020/02/20 13:08:06
ac1420215df2ac3604C7, Accept, 2020/02/20 13:08:06
ac1420215df2ac3604C7, Finish, 2020/02/20 13:08:54
ac1420215df2ac3704C9, Accept, 2020/02/20 13:08:07
ac1420215df2ac3704C9, Finish, 2020/02/20 13:08:55
```



Example: Report differences between event log records in a SQLite database versus an Oracle database:

```
# pblog --db -f /var/log/pb.eventlog.db -O oracle -D
uniqueid, etype, epoch
ac1420215df2ac6d04EC, Reject, 2020/02/20 13:09:01
ac1420215df2ac6d04EE, Accept, 2020/02/20 13:09:01
ac1420215df2ac6d04EE, Finish, 2020/02/20 13:09:49
ac1420215df2ac6d04F1, Accept, 2020/02/20 13:09:01
ac1420215df2ac6d04F1, Finish, 2020/02/20 13:09:49
```

Dumping Records in JSON Format

Starting with v10.3.0, the option **-J** has been added to display the event log records in JSON format. Combine with **-P** to enhance readability.



Example:

```
# pblog --db -f /var/log/pb.eventlog.db -J -P
```



For more information, please see "[pbmasterd](#)" on page 441.

pblogd

pblogd is the log server daemon that records event and I/O logs as directed by other Privilege Management for Unix and Linux programs. A socket-listener process (typically **inetd**, **xinetd**, or **pblogd -d**) starts **pblogd**.

Changes that are made to the **pb.settings** file after the daemon is started do not affect the operation of the daemon. If you change the **pb.settings** file, then you must restart the daemon for the changes to take effect. If you do not restart the daemon, then the daemon continues to operate using a snapshot of the **pb.settings** file that is cached at the time the daemon is started.


Syntax

- **Version 3.5 and earlier:** long command options not supported.
- **Version 4.0 and later:** long command options supported.

```
pblogd [options]
-a, --syslog_accepts
-d, --daemon
-D, --debug=<level>
-e, --error_log=log_file_name
-f, --foreground
-i, --info <argument placeholder characters>
-p, --port=port_number
-r, --syslog_rejects
-s, --syslog
pblogd -v | --version
pblogd --help
```

Arguments

-a, --syslog_accepts	Optional. Records accepted task requests in the syslog.
-d, --daemon	Optional. Runs as stand-alone daemon instead of from inetd or xinetd . This mode listens to the port that is defined in the logdport setting or the -p command line argument.
-D, --debug=<level>	Optional. Generate debug trace logs in the same directory pointed to by pblogdlog . Version 7.5 and earlier: option not available. Version 8.0 and later: option available.
-e, --error_log=log_file_name	Optional. Records diagnostic messages in the file logfile .
-f, --foreground	Optional. Run in the foreground when using daemon mode.
-i, -info <argument placeholder characters>	On Linux and AIX, the pblogd process replaces the argument placeholder characters with the following information about the submitting request: <ul style="list-style-type: none"> • submitting user • submit host • pbrun's pid • runuser

<ul style="list-style-type: none"> • runargv 	<p>The format is:</p> <pre style="background-color: #f0f0f0; padding: 5px;">submituser@submithost pid runuser: runargv</pre> <p>This allows an administrator to use the ps command to view more information about the running pblogd processes.</p> <div style="border: 1px solid #0070c0; background-color: #e1f5fe; padding: 5px; margin-top: 10px;">  Note: This feature is not available on HP-UX and Solaris. </div>
-p, --port=port	Optional. When running as a stand-alone daemon, listens to the provided port instead of the default.
-r, --syslog_rejects	Optional. Records rejected task requests in the syslog (unless Privilege Management for Unix and Linux is configured with a log server).
-s, --syslog	Optional. Records diagnostic messages in the syslog. -s works only if /etc/syslog.conf is configured to have syslog process auth.err (or less severe) messages.
-v, --version	Optional. Displays the program version and exits.
--help	Optional. Displays the program help message and exits.

Files

/etc/pb.settings	Privilege Management for Unix and Linux settings file
-------------------------	---



For more information, please see the following:

- ["pbcheck" on page 371](#)
- ["pbhostid" on page 425](#)
- ["pbkey" on page 426](#)
- ["pbmasterd" on page 441](#)
- ["pblocald" on page 430](#)
- ["pbpasswd" on page 446](#)
- ["pbreplay" on page 454](#)
- ["pbrun" on page 465](#)
- ["pbsum" on page 471](#)

pbmasterd

The policy server daemon, **pbmasterd**, is the Privilege Management for Unix and Linux decision-maker. **pbmasterd** receives secured task requests from **pbrun**, **pbguid**, **pbksh**, and **pbsh** and evaluates them according to the policy that is written in the configuration file that is specified in the settings file or `/opt/pbul/policies/pb.conf`. If the request is accepted, then **pbmasterd** directs either the client or **pblocald** to run the request in a controlled account such as root.

Policy server daemons should reside on a secure machine and are started from a socket-listener process (typically **inetd**, **xinetd**, or **pbmasterd**).

pbmasterd expects to find the configuration file in the **policyfile** setting in the settings file (default `/opt/pbul/policies/pb.conf`) on the host where **pbmasterd** resides. There may be more than one policy server daemon on different hosts for redundancy or to serve multiple networks.

pbmasterd logs all diagnostic messages in a log file that is specified by the **pbmasterdlog** setting or the **-e** command line argument.

Changes that are made to the **pb.settings** file after the daemon is started do not affect the operation of the daemon. If you change the **pb.settings** file, then you must restart the daemon for the changes to take effect. If you do not restart the daemon, then the daemon continues to operate using a snapshot of the **pb.settings** file that was cached at the time the daemon was started.


Syntax

- **Version 3.5 and earlier:** long command options not supported.
- **Version 4.0 and later:** long command options supported.

```
pbmasterd [options]
  -a, --syslog_accepts
  -d, --daemon
  --disable_optimized_runmode
  -D, --debug=<level>
  -e, --error_log=log_file_name
  -f, --foreground
  -i, --info <argument_placeholder_characters>
  -p, --port=port
  -r, --syslog_rejects
  -s, --syslog
  -V, --check_version
pbmasterd -v | --version
pbmasterd --help
```

Arguments

-a, --syslog_accepts	Optional. Sends job accept messages to syslog (unless Privilege Management for Unix and Linux is configured with a log server).
-d, --daemon	Optional. Runs as a stand-alone daemon instead of from inetd or xinetd . This mode listens to the port that is defined in the masterport setting or by the -p command line argument.
--disable_optimized_runmode	Optional. Disable pbrun optimization and use pblocald even when submit host and run host are the same. Version 4.0 and earlier: option not available.

	Version 5.0 and later: option available.
-D, --debug=<level>	Generate debug trace logs in the same directory pointed to by pbmasterdlog . Version 7.5 and earlier: option not available. Version 8.0 and later: option available.
-e, --error_log=log_file_name	Optional. Records diagnostic messages in the file logfile.
-f, --foreground	pbmasterd normally spawns a child process and dissociates from the job that starts it. Although this method is beneficial when running from inetd , xinetd , or the command line, it stops pbmasterd from running under the init daemon (from /etc/inittab). This switch prevents pbmasterd from dissociating and allows it to run from the inittab .
-i -i-info <argument placeholder characters>	On Linux and AIX, the pbmasterd process replaces the argument placeholder characters with information about the submitting request. Prior to evaluating the policy, this includes the submitting user, submit host, pbrun 's pid, the word EVALUATING , requestuser, requested argv. After evaluating the policy, this is changed to include submitting user, submit host, pbrun 's pid, runuser, runargv. This allows an administrator to use the ps command to view more information about the running pbmasterd processes.
 Note: This feature is not available on HP-UX and Solaris.	
-p, --port=port	Optional. When running as a standalone daemon, listen to the provided port instead of the default.
-r, --syslog_rejects	Optional. Sends job reject messages to syslog (unless Privilege Management for Unix and Linux is configured with a log server).
-s, --syslog	Optional. Sends diagnostic messages to syslog.
-V, --check_version	Optional. Records diagnostic messages if the connecting client's version does not match the pbmasterd version.
-v, --version	Optional. Displays the program's version and exits.
--help	Optional. Displays the program's help message and exits.

Files

/opt/pbull/policies/pb.conf	Privilege Management for Unix and Linux configuration file.
------------------------------------	---



For more information, please see the following:

- ["pbcheck" on page 371](#)
- ["pbguid" on page 422](#)
- ["pbhostid" on page 425](#)
- ["pbkey" on page 426](#)

i

- *"Enable Debug Trace Logging for pbsh and pbksh" on page 363*
- *"pblocald" on page 430*
- *"pbpasswd" on page 446*
- *"pbprint" on page 450*
- *"pbreplay" on page 454*
- *"pbrun" on page 465*
- *"pbsum" on page 471*

pbmg

Version 4.0.0 and later: **pbmg** setting available.

The **pbmg** editor is similar to the **mg** editor. **mg** is a small version of GNU **emacs** with GNU-style **emacs** key bindings. The **pbmg** version has been modified so that it can be used securely with the Privilege Management for Unix and Linux programs. Security is enhanced by the following features:

- **pbmg** must be started with a full path name specified for the file to be opened.
- The user cannot access any files other than the one that is specified at startup time.
- The user is not allowed to spawn any processes.

This program, when used with Privilege Management for Unix and Linux, allows users to access a specific file as **root**, but not other **root** functions or files.

Syntax

```
pbmg fullpathname
```

Arguments

fullpathname

File to edit.

Files

None



Example: Display the contents of the *fullpathname* file for editing:

```
pbmg fullpathname
```



For more information, please see "[pbrun](#)" on page 465.

pbnvi

Version 4.0.0 and later: pbnvi setting available.

The **pbnvi** editor is similar to the standard vi editor. It has been modified so that it can be used securely with the Privilege Management for Unix and Linux programs. The user cannot access any files other than the ones that are specified at startup time. The user is also not allowed to spawn any processes.

This program, when used in conjunction with Privilege Management for Unix and Linux, can allow users to access a specific file as root, but not access other **root** functions or files.

The edited file is written back to the same path. If this path has been changed by an external process, then the file is written to the new location to which the path now refers. Whenever **pbnvi** is run from Privilege Management for Unix and Linux, the arguments should be checked to ensure that the user could not change the path and introduce a security hole.

Syntax

```
pbnvi fullpathname
```

Arguments

fullpathname	File to edit.
---------------------	---------------

Files

None



Example: Display the contents of the fullpathname file for editing:

```
pbnvi fullpathname
```



For more information, please see "[pbrun](#)" on page 465.

pbpasswd

Version 10.3.0: **pbpasswd** deprecated. **pbpasswd** functionality is supported through the **pbadmin** program.

pbpasswd generates an encrypted password that can be used by the **getstringpasswd()** function in the configuration file. When you run **pbpasswd**, you can specify a password with the **-i** option or provide it on standard input. If standard input is a tty, then the program asks you to type the password twice; otherwise, it reads the raw password from standard input.

The program then writes the encrypted version of the password to the file that is specified by the **-o** option, or prints it on standard output.

Syntax

- **Version 3.5 and earlier:** long command options not supported.
- **Version 4.0 and later:** long command options supported.

```
pbpasswd [options]
  -i, --password=password
  -o, --output=filename
pbpasswd -v | --versions
pbpasswd --help
```

Arguments

-i, --password=password	Optional. The password to encrypt.
-o, --output=filename	Optional. Writes the encrypted version of the password to the specified output file.
-v	Optional. Displays the pbpasswd version and exits.
--help	Displays the program help message and exits.

Files

None



Example: Encrypts the string **FELIX** and place the encrypted result in the file **/tmp/pbpasswd.out**:

```
pbpasswd -i FELIX -o /tmp/pbpasswd.out
```



Example: In this example, the string **FELIX** is hashed and displayed. The first line shows the command that was entered and the second line shows the hashed string.

```
pbpasswd -i FELIX
0123m68dg5.87
```



Example: This example prompts for a new password, encrypts it, and displays the result in the standard output. The first line shows the command `cat` was entered, and the next two lines prompt for the password and prompt for the password to be retyped. The final two lines display the result of the hashed string.

```
pbpasswd
Enter Password:
Retype Password:
The encrypted password is:
4567j68gf5.88
```



For more information, please see the following:

- "[pbcheck](#)" on page 371
- "[pbmasterd](#)" on page 441
- "[pbrun](#)" on page 465

pbping

- **Version 6.2 and earlier:** **pbping** not available.
- **Version 7.0 and later:** **pbping** available.

The **pbping** program is BeyondTrust Privilege Management’s client connectivity health check utility. **pbping**, run from a policy server daemon, checks connectivity to licensed clients’ **pblocald** daemon. When run without options, **pbping** checks connectivity to all licensed clients.

Syntax

```
pbping [options]
  -h, --host host_name|IP_address
  --csv
pbping -v, --version
pbping --help
```

Arguments

-h, --host host_name IP_address	Checks a single host by host name or IP address.
--csv	Produces output in comma-separated values.
-v, --version	Displays the program information and exits.
--help	Displays the program help message and exits.

Files

None



Example: Checks the status of all client hosts and return the results in comma-separated values format:

```
pbping --csv
```



Example: Checks the status of **host1**:

```
pbping -h host1:
```



For more information, please see the following:

i

- *"pbcheck" on page 371*
- *"pblockd" on page 430*
- *"pbmasterd" on page 441*

pbprint

The **pbprint** program is used to format a configuration file. It reads the specified file and sends the formatted information to standard output. **pbprint** ignores incoming white space and places white space, indentation, and line breaks in the appropriate places. Except for line breaks that are added at the ends of statements, the program attempts to preserve line breaks from the original file as much as possible.

Syntax

- **Version 3.5 and earlier:** long command options not supported.
- **Version 4.0 and later:** long command options supported.

```
pbprint [options]
  -f, --input=file_name
pbprint -v | --version
pbprint --help
```

Arguments

-f, --input=<filename>	Specifies the policy configuration file to use.
-v, --version	Displays the program information and exit.
--help	Displays the program help message and exit.

Files

None



Example: Print the policy file:

```
pbprint
```



Note: If the installation is prefixed and/or suffixed, then the prefixed and/or suffixed configuration file is printed.



Example: Print the policy file at the specified path:

```
pbprint -f <pathtofilename>
```



Example: Print the policy file at the specified path:

```
pbprint --input=<pathtofilename>
```



For more information, please see "[pbcheck](#)" on page 371.

pbregister

Registers a Privilege Management for Unix and Linux client or secondary server to the primary license server.

The command line utility provides a method of retrieving default configuration and required data files from the primary license server to aid the initial install of the service.



Note: *Pbregister* is primarily used by Privilege Management for Unix and Linux installer. We highly recommend that its direct use should be made under the guidance of BeyondTrust Technical Support.

Syntax

- **Version 9.0 and earlier:** **pbregister** options not supported.
- **Version 10.0.1 and later:** **pbregister** options supported.

Usage

```
pbregister <options...>
```

Arguments

-a <application id>	The REST Application ID generated on the primary license server. Required to authenticate against the remote REST service.
-k <application key>	The REST Application Key generated on the primary license server. Required to authenticate against the remote REST service.
-n <hostname/address>	The TCP/IP address/domain name to contact the primary license server REST service. This is required.
-p <port>	The TCP/IP port to contact the primary license server REST service. If not specified, default is 24351.
-c <path>	Path to the libcurl shared libraries supplied by Privilege Management for Unix and Linux. Multiple libraries can be specified separated by a colon.
-s <path>	Path to the libcom_err and libssl shared libraries supplied by Privilege Management for Unix and Linux. Multiple libraries can be specified separated by a colon.
-N <profile>	The Registration Client Profile name, as defined in the primary license server database. If none is supplied the default profile is used.
-P <prefix>	The Privilege Management for Unix and Linux prefix.
-S <suffix>	The Privilege Management for Unix and Linux suffix.
-z	Set this hosts name when registering with Registry Name Service.
-g '{"svcname" : "<name>" ,	Register this host as given roles within specified service groups.

<code>"role" : "<primary secondary client>", {...}]'</code>	
<code>-R</code>	Register with the Primary Registry Name Service.
<code>-C</code>	Create simple x509 certificate.
<code>-q</code>	Run the command in <i>quiet</i> mode. Displays minimal feedback, and will not interact.
<code>-v</code>	--version

pbreplay

pbreplay replays the contents of an I/O log file for an active or completed session.

If you set the **iolog** variable to a unique path name in a policy file, then Privilege Management for Unix and Linux logs all of the input and output from the secured task to an I/O log file.

The input and output for the secured task can be logged to a file in **/usr/adm** with a file name, such as **pb.jqpublic.ksh.a05998**, that can be examined later using **pbreplay**. The name of the I/O log is a unique temporary file name that is generated by the **logmktemp()** function in the configuration file.



Example: This is an example of such a filename:

```
iolog=mktemp("/usr/adm/pb." + user + "." + basename(command) + ".XXXXXX");
```



Note: Privilege Management for Unix and Linux sets the permissions on the I/O log file so that only **root** can read the file. No other user can examine the contents of the I/O log files. You must be logged in as **root** to use **pbreplay** on these files. You can also use Privilege Management for Unix and Linux to delegate this privilege to the appropriate people.

Starting in version 9.4.0, when a user requests to replay an archived IO log, **pbreplay** makes a **REST GET** request for a copy of the archived logfile from the Log Archive Storage Server. The copy of the file is saved to a temporary file for use by **pbreplay**. When **pbreplay** exits, this temporary file is removed.

pbreplay has four main ways of replaying an **iolog**:

- **Interactively:** The viewer interactively controls the speed of viewing the **iolog**.
- **Raw:** The entire **iolog** is replayed at *computer speed*. The raw terminal control codes are interpreted by the terminal (resulting in visually correct output that is not easily searched via **grep**).
- **Processed:** The entire **iolog** is replayed at *computer speed*. The terminal control codes are interpreted by **pbreplay** (resulting in output that is searchable).
- **Policy variables:** The values of Privilege Management for Unix and Linux policy variables are printed.

Interactive Syntax

```
pbreplay [ -ao | -ax ] [ -h ] [ -t <date format> ] <iolog filename>
```

Raw Syntax

```
pbreplay <[ -I | -i | -o | -e ]> [ [-ao | -ax | -am ] ] [ -h ] [ -m ] [ -t <date format> ]
```

Processed Syntax

```
pbreplay -O [--regex <regex expression> [--ignore-case ] [ -c <constraint expression> ] [ -p <format expression> ] ] <iolog filename>
```

```
pbreplay -O [--regex <regex expression> [--ignore-case ] [ -c <constraint expression> ] [ -p <format expression> ] --files <glob pattern>]
```

Solr or Elasticsearch Indexing Syntax

```
pbreplay -X [-T]
pbreplay -Z [-T]
```

Policy Variable Only Syntax

```
pbreplay -av <iolog filename>
```

The **-O** option by itself produces searchable output by processing the terminal control codes in a virtual screen, then produces output based on that virtual screen. This feature is intended to search shell sessions delegated by **pbrun**, for shell command entries. Searching for data within a CURSES application is not supported. The goal of being able to search for shell commands that the user enters is problematic. **stdin** is not used, since it can contain shell history recall and editing commands that the shell can process, but for which Privilege Management has no context. This mechanism works on the **stdout** data stream which should normally contain the resulting command with editing completed. This is easily defeated, however, by turning echo off.

This has several limitations:

- This means that the output has newline characters when the screen width is reached, which does not accurately represent a shell command entry that wraps across the screen from one line to the next.
- Typeahead is another problem. If the user begins typing the command before the shell prompt is output, the first few characters of the command entry are not contiguous with the rest of the command.
- Termcap data for a given terminal is actually different across different platforms. For example, the vt100 left arrow is very different between Solaris and HP-UX. It may be necessary to replay I/O logs on the same operating system as they were generated from.
- Large window sizes require more memory and processing time.
- For I/O logs that have no TERM environment variable (cron jobs, for example), the xterm TERM is used.
- Terminal commands that do not directly affect the screen data or the cursor position are ignored (for example: reverse video, dim, bright, blinking, underlined).

Given the limitations, this output is suitable for processing via **grep**, **awk**, **perl**, etc. If initial searches fail to find the offending pattern, try searches allowing for typeahead, and try searching for commands that turn echo off.

The **--regex** option enables built-in searching via the standard regcomp mechanism.

This uses POSIX Extended Regular Expression syntax. Substring addressing of matches is not supported. The match-any-character operators don't match a newline. The match-beginning-of-line operator (^) matches the empty string immediately after a newline, and the match-end-of-line operator (\$) matches the empty string immediately before a newline.



Note: Since this feature works on output (not `stdin`), the beginning of line for command entry is typically a shell prompt.

The following options require both `-O` and `--regex`:

- `--ignore-case`: Ignores case during the regex comparison.
- `-c <constraint expression>`: Allows the search to be limited to I/O logs whose policy variables meet the criteria specified in the constraint expression (for example, the search can be limited to I/O logs for a specific runhost, or specific submituser).
- `-p <format expression>`: Allows the output to be customized.
- `--files <glob pattern>`: Allows multiple files to be searched. The glob mechanism supports the question mark symbol (?) matching any single character, the asterisk (*) matching any string, and character classes, ranges, and complementation within square brackets ([and]).
- The `-X` option processes the terminal control codes in the same way as `-O`, then sends the data to Solr or Elasticsearch for indexing. Whether the data is sent to Elasticsearch or Solr depends on whether a correct Elasticsearch or Solr configuration is configured in `/etc/pb.settings`. If both Elasticsearch and Solr are configured correctly, then Elasticsearch takes precedence and data will only be sent there.
- The `-Z` option processes the terminal control codes in the same way as `-O` and `-X`, and creates XML data, without sending that data to Solr. The JSON message can be displayed on the console. If `-Z` is set and `/etc/pb.settings` is correctly configured to send iologs to Elasticsearch, then the data is rendered on the console in JSON format. Otherwise, if `/etc/pb.settings` is configured correctly for Solr, then the data is rendered in XML format. As with `-X`, if both Elasticsearch and Solr are configured correctly, then Elasticsearch takes precedence and the data is presented on the console in JSON.

Syntax

- **Version 3.5 and earlier:** long command options not supported.
- **Version 4.0 and later:** long command options supported.


```
pbreplay [options] I/O_log_name
  --history
  --history2
  --history3
  --hideshellstartup
  --nolinenumbers--markshellstartup
  --showall
  -am, --map_printable
  -ao, --map_octal
  -av, --variables
  -ax, --map_hex
  -aX
  -e, --show_stderr
  -h, --header
  -i, --show_raw_stdin
  -I, --show_translated_stdin
  -k, --keyfile=key_file
  -m, --more
  -o, --show_stdout
  -A --audit
  -t, --timestamp[=format]
pbreplay -O [options] [I/O_log_name]
  -R, --regex <regular expression>
  --files <file glob pattern>
```



```

-c <constraint expression>
-p <format expression>
pbreplay <--sendindex | -X> [options] [I/O_log_name]
pbreplay <--index | -Z> [options] [I/O_log_name]
pbreplay --forward
pbreplay -v | --version
pbreplay --help
    
```

Arguments

--history	Derives shell history from an ACA I/O log where session history was enabled via enablesessionhistory(true) ;
--history2	Create more detailed ACA Session History Report.
--history3	Create ACA Session History Report of all execs.
--hideshellstartup	<p>Version 10.2.0 and earlier: --hideshellstartup not available.</p> <p>Version 10.3.0 and later: --hideshellstartup available.</p> <p>Use alone, or in combination with the --history, --history2, and --history3 options.</p> <p>Commands executed as part of the shell startup scripts (identified as processes with the same process group as the main shell) are omitted from the history report.</p>
--markshellstartup	<p>Version 10.2.0 and earlier: --markshellstartup not available.</p> <p>Version 10.3.0 and later: --markshellstartup available.</p> <p>Use alone, or in combination with the --history, --history2, and --history3 options.</p> <p>Commands executed as part of the shell startup scripts (identified as processes with the same process group as the main shell) are included in the history report with the text shellstartup preceding the command and arguments.</p>
--markshellstartup	<p> Example:</p> <pre> [root@dev-01 tmp]# /tmp/pbreplay --history --markshellstartup iolog.CbCezgQDCBwV7F18tdVPAdi1 1 Info working directory set to: /tmp 2 Allowed shellstartup id -gn 3 Allowed shellstartup id -un 4 Allowed shellstartup uname -s 5 Allowed shellstartup ls /etc/bash_completion.d 6 Allowed shellstartup /bin/grep -q /home/bginn/.cabal/bin 7 Allowed shellstartup /usr/bin/tty -s 8 Allowed shellstartup /usr/bin/tput colors 2> /dev/null 9 Allowed shellstartup /usr/bin/dircolors --sh /etc/DIR_COLORS 2> /dev/null 10 Allowed shellstartup /bin/grep -qi ^COLOR.*none /etc/DIR_COLORS > /dev/null 2> /dev/null 10 Allowed shellstartup /usr/bin/id -u 11 Allowed shellstartup /bin/hostname 12 Allowed shellstartup sed -e s#\..*\$## 12 Allowed shellstartup /bin/hostname </pre>

	 <pre> 13 Allowed shellstartup head -1 13 Allowed shellstartup tail -1 13 Allowed shellstartup sed -e s#\.*\$## 13 Allowed shellstartup head -1 13 Allowed shellstartup tail -1 13 Allowed shellstartup /usr/bin/tty 14 Allowed shellstartup sed -e s#/#g 15 Allowed id 17 Allowed date [root@dev-01 tmp]# </pre>
--nolinenumbers	Do not display line numbers on the ACA history report.
--showall	Do not remove duplicate output from the ACA Audit report.
-am, --map_printable	Optional. Maps unprintable characters in the selected streams (-e , -i , -l , -o) to printable sequences.
-ao, --map_octal	Optional. Maps unprintable characters in the selected streams to octal format. (\xxx).
-av, --variables	Optional. Displays the variables for the secured task.
-ax, --map_hex	Optional. Maps unprintable characters in the selected streams to hexadecimal format.
-aX [mV], --map_xwindows	Optional. Dumps relevant X11 captured events from the iolog . Major events such as creating and destroying windows, textual window updates, text input and mouse clicks are displayed as a summary alongside any output from the parent process. m adds selected streams, V logs all X events.
-e, --show_stderr	Optional. Non-interactive. Dumps the contents of the standard error stream. This option is useful for redirecting the data to a file or another program. For example, the output can be piped to grep to search for specific words or sequences.
-h, --header	Optional. Non-interactive. Prints a header that contains the secured task information. This option is useful only with dump options. The header is always printed in interactive mode.
-i, --show_raw_stdin	Optional. Non-interactive. Dumps the contents of the standard input stream. This option is useful for redirecting the data to a file or another program. For example, the output can be piped to grep to search for specific words or sequences.  Note: If the user typed carriage returns during the session, the input stream might contain carriage returns without linefeeds. On replay these cause the cursor to return to the left margin of the display and overwrite the previous line with the next line. If you are examining the data, lines may appear to be lost. To add line feeds to the display, use the option: -l (--show_translated_stdin) .
-l, --show_translated_stdin	Optional. Non-interactive. Dumps the contents of the standard input stream, adding linefeeds to carriage returns to improve readability. This option is useful for redirecting the data to a file or another program. For example, the output can be piped to grep to search for specific words or sequences.

-k, --keyfile=<name>	Optional. Uses the named key file to decrypt an encrypted I/O log file.
-m, --more	Optional. Interactively display the I/O log file using a pager-styles display (for example, less , more , pg) instead of using the native display.
-o, --show_stdout	Optional. Non-interactive. Dumps the raw contents of the standard output stream. The terminal interprets the raw terminal commands such as <beginning of line> , <delete char> , and <delete word> . This option is useful for viewing the logged output.
-A, --audit	Display ACA audit stream.
-t, --timestamp [=format]	Displays timestamps on each line of output. The optional format argument can be any format string that is suitable for the date command. If the optional format string is omitted, then the system -t, --timestamp[=format] default time format is used. [ver 4.0 and earlier]: option not available. [ver 5.0 and later]: option available.
-v, --version	Required. Displays the program's version and exits.
--help	Optional. Displays the program's help message and exits.

The ACA Audit report lists file related libc/system calls and whether those calls were allowed or blocked via ACA.

Prior to Privilege Management for Unix and Linux 9.4, the output was similar to:

```
Fri Nov 11 12:02:41 2016:    7115    1    owner    9    execve /usr/bin/id
Fri Nov 11 12:02:41 2016:    7115    1    read     9    fopen  /etc/passwd
Fri Nov 11 12:02:41 2016:    7115    1    read     9    fopen  /etc/group
```

Starting with PMUL 9.4, the output is changed to:

```
Fri Nov 11 2016 12:02:41 PM [ 7115] Allowed exec /usr/bin/id
Fri Nov 11 2016 12:02:41 PM [ 7115] Allowed read /etc/passwd
Fri Nov 11 2016 12:02:41 PM [ 7115] Allowed read /etc/group
```

And with additional loglevels:

```
Fri Nov 11 2016 12:08:36 PM [ 7359] Allowed read /etc/group TAG:DEFAULT dev:64768 ino:2107740
mode:100644 uid:0 gid:0
Fri Nov 11 2016 12:08:37 PM [ 7368] Allowed exec /usr/bin/head ARGV:[head /etc/passwd]
TAG:DEFAULT
ENV: LOGNAME=jsmith
ENV: PWD=/home/jsmith
ENV: HISTSIZE=1000
```


The ACA history report derives shell history from the ACA I/O log, and produces a report similar to a shell's history command:

```
# pbreplay --history aca.iolog.log.aZTkfJ
1 Info      working directory set to: /home/jsmith
2 Allowed   date
```

```
3 Allowed id
4 Allowed head /etc/passwd
```

-O Optional. Non-interactive. Produces searchable output by interpreting tty commands such as **<beginning of line>**, **<insert char>**, and **<delete word>**. This option is useful for redirecting the data to a file or piping to another program. For example, the output can be piped to **grep** to search for specific words or sequences. This option also allows direct searching via the **--regex** option.

-O Arguments

--R, --regex <regular expression>	Optional regular expression to search for using built-in search mechanism. The default output includes runhost, user, date/time, and matching line of text.
--ignore-case	Optional flag to ignore case during regex search.
-c <constraint expression>	Optional constraint expression using the policy language. Similar to constraints used by pblog . Requires the -O and --regex options.
-p <format expression>	Optional output format expression using the policy language. Similar to format expressions used by pblog . Requires the -O and --regex options.
--files "<file glob pattern>"	Optional file glob pattern used to search more than one file at a time. <div style="border: 1px solid black; background-color: #e0f0ff; padding: 5px; margin: 5px 0;">  Note: Quotes are required, so that the shell does not interpret the glob pattern prior to passing it to pbreplay. </div> Requires the -O and --regex options. The default output includes filename, runhost, user, date/time, and matching line of text.

-X Arguments

-T, --indextime	Include newline timestamp in indexed data.
-Q	Retrieve filenames from indexing queue.

-Z Arguments

-T, --indextime	Include newline timestamp in indexed data.
--forward	Index IOLogs identified in the IOLog index store and forward file.

The following table shows the keyboard keys that can be used with **pbreplay** in interactive mode to emulate the Unix/Linux pager:

Keyboard Keys Used with pbreplay in Interactive Mode

Key	Description
g	Go to the beginning of the I/O log.

Key	Description
G	Go to end.
Space	Display the next screen of the I/O log.
<CR> or <NL>	Advance the display one line.
s	Skip to the next time marker.
u	Undo.
t	Display the time stamp from the current line in the I/O log file.
r	Redraw from the start.
q or Q	Quit.
v	Display the variables for the I/O log file.
Backspace or Delete	Back up to the last position.
c	Continuous slow speed replay.
+ or -	Use + to increase the replay speed. Use a hyphen (-) to decrease the replay speed. Version 5.1.1 and earlier: option not available. Version 5.1.2 and later: option available.
f	Find.
k	Find time stamp. The format is [MM/DD/[CC]YY HH:M [:SS] .
.h or ?	Display a navigation help message.

Files

I/O log file



Example:

```
pbreplay /usr/adm/pb.jqpublic.ksh.a05998
```

Entering the command above produces output similar to the following:

```
Start of log =====
2005/09/08 15:16:07 jqpublic@athena.unix.company.com -> root@athena.company.com ksh
Commands:
g - go to start,
G - go to end,
Space - go to next input
<CR> or <NL> - go to next newline,
```



```

s - skip to next time marker
u - undo,
t - display time, r - redraw from start, q/Q - quit
v - dump variables, <BS> or <DEL> - backup to last position
c - continuous slow speed replay, f - find, k - find time stamp
h or ? - display this help message
    
```

You can navigate the I/O log file by pressing the space key (next input character), the carriage return or newline key (newline), or the **s** character, which shows you what happened each second. Alternatively, you can back up through the log file by pressing the **Backspace** or **Delete** key. You can quickly go to the start or end of the log file using **g** or **G**, respectively. Display the time of an action at any point in the log file using **t**, redraw the log file using **r**, and undo your last action using **u**. You can also display all of the environment variables that were in use at the time the log file was created using **v**. Use **q** or **Q** to quit **pbreplay**.


Example:

```

pbreplay -O /var/log/pbul/iolog.aaaaaa
pbreplay -O --regex 'passwd' /var/log/pbul/iolog.aaaaaa
pbreplay -O --regex 'passwd' --files '/var/log/PBLOGS*/iolog.*'
pbreplay -O --regex "passwd" --files "/var/log/pbul.iolog.*" \
    -c "runhost=='hostabc.beyondtrust.com'" -c "date=='2012/04/10'"
pbreplay -O --regex 'passwd' --files '/var/log/PBLOGS*/iolog.*' \
    -p "sprintf('%s %s %s %s\n', basename(iolog), user, runhost, regexmatch)"
    
```



For more information, please see the following:

- **man regcomp**
- **man glob(7)**
- ["pbcheck" on page 371](#)
- ["pbguid" on page 422](#)
- ["pbhostid" on page 425](#)
- ["pbkey" on page 426](#)
- ["pblocald" on page 430](#)
- ["pbmasterd" on page 441](#)
- ["pbpasswd" on page 446](#)
- ["pbrun" on page 465](#)
- ["pbsum" on page 471](#)

pbreport

- **Version 3.5 and earlier:** **pbreport** not available.
- **Version 4.0 and later:** **pbreport** available.

pbreport extracts data from the Privilege Management for Unix and Linux event logs and optionally generates one or more reports based on the extracted data as specified in the named report file. The report file is an XML file that contains the extraction and reporting specifications, and is typically configured using the event log reporting feature of **pbguid**, the Privilege Management for Unix and Linux Web GUI.

Syntax

```
pbreport [options] report-file
-i, --info
-k, --keyfile=file_name
-s, --suppress
-t, --temp
-V, --verbos
pbreport -v | --version
pbreport -h | --help
```

Arguments

-i, --info	Optional. Displays detailed information about the named report-file.
-k, --keyfile=file_name	Optional. Name of the encryption key file.
-s, --suppress	Optional. Suppress report output to the screen and user-defined commands, if any, that are specified in the named report-file. Useful for testing report runs by overriding any user-defined commands that are specified in the report specifications.
-t, --temp	Optional. Does not remove the temporary files that are created during the reporting process. If reporting problems occur, then you can send these files to BeyondTrust Technical Support for analysis.
-V, --verbose	Optional. Displays more verbose output.
-v, --version	Optional. Displays the program version and exits.
-h, --help	Optional. Displays the program help and usage information.

Files

Report set file



Example:

```
pbreport /temp/report.set
```

i For more information, please see the following:

- "[pbguid](#)" on page 422
- "[pblog](#)" on page 433

pbrun

pbrun requests that a secure task be run in a controlled environment. The user prefixes the command line with **pbrun**.

Example:

```
pbrun backup /usr/dev/dat
```

pbrun checks the settings file for a **submitmasters** entry or the netgroup **@pbsubmitmasters** to determine the policy server daemon to which it should send the request. If the policy server daemon accepts the request, then it directs a local daemon to start the task request on the run host.

Syntax

- **Version 3.5 and earlier:** long command options not supported.
- **Version 4.0 and later:** long command options supported.

```
pbrun [options] command [command_arguments]
-b, --background
-d, --debug=connect
-d, --debug=log=<level>
-d, --debug=mlog=<level>
-d, --debug=glog=<level>
-d, --debug=llog=<level>
-d, --debug=time
-d, --debug=ttime
--disable_optimized_mode
-h, --host=run_host
-l, --local_mode
-n, --null_input
-p, --pipe_mode
--solarisproject projectname
-u, --user=request_user
--testmaster=master_host
-X
pbrun -v | --version
pbrun --help
```

Arguments

-b, --background	Optional. The target job is directed to ignore hang up signals. This option is particularly useful for running the target program in the background.
-d connect, --debug=connect	Optional. Displays policy server connection information for debugging.
-d log=level, --debug=log=level	Optional. Generate debug trace logs for pbrun and all active Privilege Management for Unix and Linux components that process the command. Specify a level number from 1 (least detail) to 9 (most detail). The resulting logs reside in the same location as the corresponding diagnostics log.

	<p>Version 7.5 and earlier: setting not available.</p> <p>Version 8.0 and later: setting available.</p>
-d glog=level, --debug=glog=level	<p>Optional and only available when running as root. Generate debug trace log for pblogd that processes the command. This setting is made permanent for that log host. Specify a level number from 1 (least detail) to 9 (most detail). The resulting logs reside in the same location as the pblogd diagnostic log file.</p> <p>Version 7.5 and earlier: setting not available.</p> <p>Version 8.0 and later: setting available.</p>
-d mlog=level, --debug=mlog=level	<p>Optional and only available when running as root. Generate debug trace log for pbmasterd that processes the command. This setting is made permanent for that policy server host. Specify a level number from 1 (least detail) to 9 (most detail). The resulting logs reside in the same location as the pbmasterd diagnostic log file.</p> <p>Version 7.5 and earlier: setting not available.</p> <p>Version 8.0 and later: setting available.</p>
-d time, --debug=time	Optional. Displays pbrun timing information for debugging. This option is intended primarily for BeyondTrust Technical Support.
-d ttime, --debug=ttime	Optional. Display pbrun total run time for debugging.
--disable_optimized_runmode	<p>Disable pbrun optimization and use pblocald even when submit host and run host are the same. This affects only the local submit host.</p> <p>Version 4.0 and earlier: option not available.</p> <p>Version 5.0 and later: option available.</p>
-h, --host=run_host	Optional. Requests run_host as the run host for the secured task. Ignored if -l is also specified, or if the runlocalmode policy variable is set to true .
-l, --local_mode	Optional. Requests that the secured task run locally. Once the policy server host's policy accepts the request and logs its start, the target program replaces the pbrun on the local machine. This option provides increased efficiency and reduced network traffic, but job termination status and timeout processing. This mode can be disabled in the configuration file by setting allowlocalmode to false. This mode can also be overridden in the policy by setting runlocalmode to 0 .
-n, --null_input	Optional. Redirects the standard input of pbrun to /dev/null . You sometimes need this option to avoid interactions between pbrun and the shell that invokes it. For example, if you are running pbrun and start pbrun in the background without redirecting its input away from the terminal, it blocks even if no reads are posted by the remote command. These options prevent this situation.
-p, --pipe_mode	Optional. Puts pbrun into pipe mode. Forces the secured task to behave as if it is run in a pipeline rather than a terminal session.
--solarisproject projectname	<p>Optional. Associates the Solaris project projectname with the secured task. Requires Solaris version 9 or later on the runhost.</p> <p>Version 6.0 and earlier: option not available.</p> <p>Version 6.1 and later: option available.</p>
--testmaster=master_host	Optional and only available when running as root. Requests master_host as the policy server host

	<p>to test whether a command will be accepted or rejected. The command itself is not executed. Specify either the hostname or the IP address for the master_host.</p> <p>Version 7.5 and earlier: option not available.</p> <p>Version 8.0 and later: option available.</p>
-u, --user=request_user	Optional. Sets the variable requestuser to request_user . The policy can then decide to honor the request and set runuser and/or runeffectiveuser equal to request_user .
-v, --version	Optional. Displays the program version and exits.
--help	Optional. Displays the program help message and exits.
-X	<p>Optional. Activates X11 forwarding.</p> <p>When running pbrun with the -X option, the DISPLAY environment variable needs to be set, and a valid XAuthority token needs to exist in the users .Xauthority file. This can be checked using:</p> <pre>xauth list \$DISPLAY</pre>

Files

/etc/pb.settings	Local Privilege Management for Unix and Linux submithost settings.
-------------------------	---



Example:

```
pbrun -h runhost uname -a
```



For more information, please see the following:

- the **pb.settings** file
- "[Debug Trace Logging](#)" on page 330
- the **xwinforward** and **xwinreconnect** policy variables in the [Privilege Management for Unix and Linux Policy Language Guide](https://www.beyondtrust.com/docs/privilege-management/unix-linux/index.htm) at <https://www.beyondtrust.com/docs/privilege-management/unix-linux/index.htm>
- "[pbcheck](#)" on page 371
- "[pbhostid](#)" on page 425
- "[pblocald](#)" on page 430
- "[pblog](#)" on page 433
- "[pbmasterd](#)" on page 441
- "[pbpasswd](#)" on page 446
- "[pbreplay](#)" on page 454
- "[pbsum](#)" on page 471

pbssh

- **Version 6.0.1 and earlier:** **pbssh** program not available.
- **Version 6.1 and later:** **pbssh** program available.

Using Privilege Management for Unix and Linux policy and the **pbssh** program, you can control access to, and activities on, SSH-managed devices. The **pbssh** program is similar to the **pbrun** program, except that it uses the SSH protocol (or, optionally, the telnet protocol) to connect to devices that do not have Privilege Management for Unix and Linux installed on them; such devices can include Windows computers and certain network devices.

You must specify the **-h** option (to indicate the host name of the target device), and the **-u** option (to indicate the user name with which to log into the device). To execute a command on the target device, use the **-C** option. You may also optionally use the **-P (--port)** option to specify a particular port for the SSH connection.

If you have a Password Safe appliance, the Privilege Management for Unix and Linux can be configured to automatically obtain the device password from Password Safe. To do so, the following Privilege Management for Unix and Linux settings must be specified on the submit host:

- **pkrunfile**
- **pk_cert** (or the **--pk_cert** option)
- **pk_servers** (or the **--pk_servers** option)
- **pbsshshell** (optional)

If you do not have a Password Safe appliance, then **pbssh** prompts the user for the password. The user is also prompted under these circumstances:

- The Password Safe appliance is not available.
- The Privilege Management for Unix and Linux settings are not specified or not correctly specified.
- The **--skip_pkrun** option is specified on the **pbrun** command line.
- The **--telnet** option is specified on the **pbrun** command line.

The **--domain** option has two purposes, both of which are related to Password Safe:

- If you need to log into a host using a domain account, then you use the **--domain** option defines the domain from which Password Safe should obtain the domain account password.
- If the **--user** option defines a user account, and you want to use a Password Safe managed account alias in place of the actual managed system name, then you use the **--domain** option to specify the managed system alias.



Note: Unlike **pbrun**, **pbssh** does not require a command to be specified. Consequently, the Privilege Management for Unix and Linux policy function **basename()** always returns **pbssh**. In the Privilege Management for Unix and Linux policy, to determine the command that was specified, parse the **argv** list.

Syntax

```
pbssh [options] command [command_arguments]
-c, --pk_cert
-C, --command
-d, --debug=connect
```

```

-d, --debug=time
-d, --debug=tttime
-D, --domain
-h, --host=run_host
-k, --skip_pk
-K, --pk_servers
-P, --port=ssh_port
-r, --pk_reset_password
-T, --telnet
-u, --user=request_user
pbssh -v | --version
pbssh --help

```

Arguments

-c, --pk_cert	Optional. Absolute path to the Password Safe certificate on the submit host. Overrides the pk_cert Privilege Management for Unix and Linux setting.
-C, --command='ssh_command'	Optional. Command and arguments to be executed on the target SSH-managed device. If arguments are specified, the command and its arguments must be enclosed together in single quotation marks.
-d connect, --debug=connect	Optional. Displays policy server connection information for debugging.
-d time, --debug=time	Optional. Displays pbssh timing information for debugging. This option is intended primarily for BeyondTrust Technical Support.
-d tttime, --debug=tttime	Optional. Displays pbssh total run time for debugging.
-D, --domain	Optional. Specifies a domain for Password Safe to use when obtaining a domain account password, or defines a Password Safe managed system alias to use instead of the actual host name. Version 6.1 and earlier: option not available. Version 6.2 and later: option available.
-h, --host=run_host	Requests run_host as the run host for the secured task.
-k, --skip_pkrun	Optional. Specifies that the SSH-managed device password not be obtained from Password Safe.
-K, --pk_servers	Optional. Specifies the host name or IP address of one or more PowerBroker Safe appliances. Overrides the pk_servers Privilege Management for Unix and Linux setting. To specify more than one PowerBroker Safe appliance, separate each name by a space and enclose the list in quotation marks.
-P, --port=ssh_port	Specifies a TCP port to use for the SSH session. If not specified, then a default port number is used.
-r, --pk_reset_password	Optional. Specifies that PowerBroker Safe check in a new password for the user after the PowerBroker Safe command is complete.
-T, --telnet	Optional. Specifies that a connection to an SSH-managed device be made using the telnet protocol, not the SSH protocol.
-u, --user=request_user	Sets the variable requestuser to request_user . The policy can then decide to honor the request and set runuser and/or runeffectiveuser equal to request_user .

-v, --version	Optional. Displays the program version and exits.
--help	Optional. Displays the program help message and exits.

Files

/etc/pb.settings Local Privilege Management for Unix and Linux **submithost** settings



Example:

```
pbssh -h runhost -u jjones -C "dir /w"
```



For more information, please see "[Connections to SSH-Managed Devices](#)" on page 95.

pbsum

pbsum prints the checksum of one or more files. The checksum can be used in a policy configuration file to check the requested program's integrity. If anyone has modified the program and thereby changes the checksum, the secured task is refused. The string that is produced by **pbsum** can be used to set the value of the **runmd5sum** variable in the Privilege Management for Unix and Linux policy configuration file.

Syntax

- **Version 3.5 and earlier:** long command options not supported.
- **Version 4.0 and later:** long command options supported.

```
pbsum file_names
pbsum -m | --md5 file_names
pbsum -v | --version
pbsum --help
```

Arguments

-m, --md5 file_names	Optional. Use the MD5 algorithm to generate a checksum.
-v, --version	Optional. Displays the pbsum version and exits.
--help	Optional. Displays the program help message and exits.

Files

None



Example:

```
pbsum /etc/pb.settings file
pbsum -m /etc/pb.settings /bin/ls
```



For more information, please see the following:

- ["pbcheck" on page 371](#)
- ["pbhostid" on page 425](#)
- ["pbkey" on page 426](#)
- ["pblocald" on page 430](#)
- ["pbmasterd" on page 441](#)

i

- *"pypasswd" on page 446*
- *"pbreplay" on page 454*
- *"pbrun" on page 465*

pbsync

- **Version 4.0 and earlier:** **pbsync** not available.
- **Version 5.0 and later:** **pbsync** available.

The **pbsync** command starts the log synchronization process. The command takes as an input one or more log servers, port numbers, and log file names, and uses that information to synchronize the network logs. This component is referred to as the client.

On the first execution of this feature, the complete event logs are transferred from the failover log servers to the primary log server; event log files are merged there into one log file to make auditing easier.

pbsync can request the following:

- That event logs from different log servers be merged
- That partial I/O logs from different log servers be merged into one I/O log
- That merged logs be sent to the client



IMPORTANT!

For encrypted I/O logs to be successfully merged, all log servers must use the same encryption algorithm and key. For more information, please see "iologencryption" on page 204.

In Privilege Management for Unix and Linux version 6.0 and later, the log synchronization server (**pbsyncd**) uses the **eventlog** setting in its own **pb.settings** file to determine the location of the event log file when it receives a **pbsync -l** request. This change can cause errors when merging pre-version 6.0 event logs if the **eventlog** setting on the log synchronization server does not match the **eventlog** setting on the requesting client host. To ensure that pre-version 6.0 event logs are found by the log synchronization server, use **pbsync** with the **-L** option.

Syntax

```
pbsync [options]
pbsync -v|--version
pbsync --help
```

Arguments

-l, --event	Uses the log synchronization server's pb.settings file to automatically obtain the event log server, port, and event log file to use. Specify the event log file name in the eventlog keyword, specify the event log server in the logservers keyword, and use the syncport keyword to specify the port.
-L, --eventlog server_info[:path/file_name[:port]]	Manually adds a server to gather the event logs from. path and file_name are optional the path and file name of the event log file to retrieve. The server_info may be a hostname or an IP address. When specifying an IPv6 address, it must be enclosed in square brackets. If port is not specified, then the default port from the settings file (syncport setting) is used. You must repeat the -L switch for each server from which you want to retrieve event log

	files.
-i, --iosearch <i>basepath</i>	<p>Queries the I/O log servers that are listed in the logservers setting in the server's pb.settings file to obtain any partial I/O log files that have basepath in their file names. basepath is the path and file name of the original I/O log file. All matching partial I/O log files are merged to create a single output I/O log file.</p> <p>Version 5.2 and earlier: option not available.</p> <p>Version 6.0 and later: option available.</p>
-I, --iolog <i>server_info:path/file_name</i> [:port]	<p>Manually adds a server to gather an I/O log file from. path and file_name are the required path and file name of the I/O log file to retrieve. The server_info may be a hostname or an IP address. When specifying an IPv6 address, it must be enclosed in square brackets. If port is not specified, then the default port from the settings file (syncport setting) is used.</p> <p>If this option is used to merge the I/O logs (without -r), only the I/O logs of the same session (partial I/O logs) are merged together in the generated output file. Any I/O log that does not belong to the same session as the first I/O log gathered are ignored. You must repeat the -I switch for each server from which you want to retrieve I/O log files.</p> <p>Version 5.2 and earlier: option not available.</p> <p>Version 6.0 and later: option available.</p>
-o, --outputfile <i>file_name</i>	User-defined path for the local output file. Cannot be used with -O or -P . Path must be to a secure directory (that is, readable and writable by root only).
-O, --outputdir <i>directory_name</i>	Uses alternate path to write the collected files. Cannot be used with -o or -P . Path must be to a secure directory (that is, readable and write able by root only).
-d, --daemon	Starts the synchronization process in daemon mode. In daemon mode, pbsync attempts to resynchronize the specified logs at a frequency that is specified by the logresyncntimermin setting in the client's pb.settings file.
-r, --retrieve	Only retrieves log files; does not merge them. Cannot be used with -P .
-v, --version	Displays the version.
--help	Displays the help message and exits.

Command Line Responses

Command Line Response	Description
Synchronization daemon unable to start	The synchronization daemon was unable to start.
Synchronization daemon: Unknown request (0xcode)	The request from pbsync was unknown or not supported.
Unable to connect to log server <name>	The system is unable to establish communications with the server, and is therefore unable to retrieve the log records.
Unable to retrieve file <path> on server <name>	The remote server reports that it is unable to read or transmit the file; check the file permissions or path.

Command Line Response	Description
Unique ID mismatch on <server name>:<path>: <uniqueID> (local ID: <uniqueID>)	The unique ID in the remote log files mismatch with the local server.
Timed out while retrieving <server name>:<path>:<port>	The operation timed out while retrieving a remote log file, causing the merge to fail.
Insufficient storage space to complete synchronization	There is insufficient storage to either retrieve or merge the file; user must free up some space.
Success	The synchronization operation was successful.

Files

None



Example: Executing the following causes **pbsync** to look for the file pattern `/var/adm/pb.user1`, collect the logs, and synchronize them:

```
pbsync -i /var/adm/pb.user1
```



Example: Executing the following synchronizes the log files on machines **dart** and **aji**:

```
pbsync -L dart:/var/log/pb.eventlog:6298 -L aji:/var/adm/pb.eventlog:6298
```



For more information, please see "[pbsyncd](#)" on page 476.

pbsyncd

- **Version 4.0 and earlier:** **pbsyncd** not available.
- **Version 5.0 and later:** **pbsyncd** available.

The **pbsyncd** server listens for log synchronization requests from one or multiple **pbsync** clients.

pbsyncd is started as a stand-alone service or daemon from the command line or startup scripts, preferably running on each system that is also running **pblogd** (for log synchronization) or **pbmaterd** (for policy updates). A new TCP port is required to accept requests. The default port is **24350**. This component is referred to as the *server*.

Log synchronization: When the server receives a synchronization request, it checks its own **pb.settings** file to determine the event log file name (as specified in the **eventlog** keyword) to retrieve. If the **eventlog** keyword is not specified, then the server uses the event log file name that is specified in the request.

Changes that are made to the **pb.settings** file after the daemon is started do not affect the operation of the daemon. If you change the **pb.settings** file, then you must restart the daemon for the changes to take effect. If you do not restart the daemon, then the daemon continues to operate using a snapshot of the **pb.settings** file that was cached at the time the daemon was started.


If **pbsyncd** detects an error, then an error message is logged in the server's diagnostic log file. For log synchronization, the client can request the following:

- That event logs from different log servers be merged
- That partial I/O logs from different log servers be merged into one I/O log
- That merged logs be sent to the client

Syntax

```
pbsyncd [options]
pbsyncd -v|--version
pbsyncd --help
```

Arguments

-d, --daemon	Runs pbsyncd in daemon mode.
	 Note: To use the -d option, you must also use the -p option.
-f, --foreground	Runs the server in the foreground when using daemon mode.
-p, --port <i>port_number</i>	Instructs the server to listen to the defined port number (use a TCP port other than the default).
-e, --errorlog <i>errorfile</i>	Uses the named file as the error log.
-s, --syslog <i><facility></i>	Uses the syslog facilities.
-v, --version	Displays the version.
--help	Displays the help message and exits.

Command Line Responses

Synchronization daemon unable to start	The synchronization daemon was unable to start.
Synchronization daemon: Unknown request (0xcode)	The request from pbsync was unknown or not supported.
Unable to connect to log server <name>	The system is unable to establish communications with the server, and is therefore unable to retrieve the log file.
Unable to retrieve file <path> on server <name>	The remote server reports that it is unable to read or transmit the file; check file permissions or path.
Unique ID mismatch on <server name>:<path>:<uniqueID> (local ID:<uniqueID>)	The unique ID in the remote log files mismatch with the local server.
Timed out while retrieving <server name>:<path>:<port>	The operation timed out while retrieving a remote log file, causing the merge to fail.
Insufficient storage space to complete synchronization	There is insufficient storage to either retrieve or merge the file; user must free up some space.
Success	The synchronization operation was successful.

Files

None



Example: Entering the following starts the daemon and specifies `/var/log/syncserver.log` as the output error file:

```
pbsyncd -d -p 24345 -e /var/log/syncserver.log
```



For more information, please see the following:

- "[pblog](#)" on page 433
- "[pbsync](#)" on page 473

pbumacs

pbumacs is an editor similar to **umacs**. **umacs** is a small version of **emacs** with gosling-style **emacs** key bindings.

pbumacs has been modified so that it can be used securely with the Privilege Management for Unix and Linux programs.

Security has been enhanced with the following features:

- **pbumacs** must be started with a full path name specified.
- The user cannot access any files other than the one that is specified at startup time.
- The user is not allowed to spawn any processes.

This program, when used with Privilege Management for Unix and Linux, can allow users to access a specific file as root, but not other **root** functions or files.

Syntax

```
pbumacs fullpathname
```

Arguments

fullpathname	File to edit.
---------------------	---------------

Files

None



Example: Executing this command displays the contents of the file called **fullpathname** for editing:

```
pbumacs fullpathname
```



For more information, please see "[pbrun](#)" on page 465.

pbufqrpq

- **Version 3.5 and earlier:** pbufqrpq not available.
- **Version 4.0 and later:** pbufqrpq available.

pbufqrpq is a licensed utility from UV Software's Vancouver Utilities set. It is used internally to generate text-based reports from a description file that is created by the Privilege Management for Unix and Linux **pbreport** program.

Syntax

```
pbufqrpq
```

Arguments

Not published

i For more information, please see the following:

- "[pblog](#)" on page 433
- "[pbreport](#)" on page 463

pbversion

- **Version 5.1.2 and earlier:** **pbversion** not available.
- **Version 5.2 and later:** **pbversion** available.

pbversion reports the binary file version numbers and optionally reports installed components of Privilege Management for Unix and Linux. This non-interactive utility can be run on any machine and provides a list of the version numbers for all known Privilege Management for Unix and Linux binary files. Installations are identified using the prefix and suffix arguments or the default installation if a prefix and/or suffix is not specified.

Only **root** can run **pbversion**. Prior to v8.0 **pbversion** can only be run from the installation directory and it should not be moved from the installation directory because it is dependent on Privilege Management for Unix and Linux installer scripts.

With v8.0 and later, **pbversion** is available as part of the installed binaries after the install (located in the admin directory which is **/usr/sbin** by default).

pbversion does not report on Privilege Management for Unix and Linux versions prior to v4.0 because the binary version arguments were not available previously. Also, it does not report on binary version for executable files **pbnvi** or **pbuvqrpq** because the version argument does not display the binary version.

Syntax

```
pbversion [options]
  -p prefix
  -s suffix
  -c
pbversion -v
pbversion -h
```

Arguments

-c	Displays the installed components.
-h	Prints usage message and exits.
-p prefix	Sets the installation prefix.
-s suffix	Sets the installation suffix.
-v	Prints the version of pbversion and exits.



Example: Display the binary versions for the installation using the prefix **my** along with the Privilege Management for Unix and Linux installation components by executing the following:

```
./pbversion -p my -c
```




For more information, please see the following:

- "pbinstall" in the *Privilege Management for Unix and Linux Installation Guide* at <https://www.beyondtrust.com/docs/privilege-management/unix-linux/index.htm>
- "pbuninstall" in the *Privilege Management for Unix and Linux Installation Guide* at <https://www.beyondtrust.com/docs/privilege-management/unix-linux/index.htm>

pbvi

- **Version 4.0.0 and later:** `pbvi` setting available.

Description

The `pbvi` editor is similar to the standard `vi` editor. It has been modified so that it can be used securely with the Privilege Management for Unix and Linux programs. Security is enhanced with the following features:

- `pbvi` must be started with a full path name specified.
- The user cannot access any files other than the one that is specified at startup time.
- The user is not allowed to spawn any processes.

This program, when used with Privilege Management for Unix and Linux, allows users to access a specific file as `root`, but not access other `root` functions or files.

The edited file is written back to the same path. If this path changed by an external process, then the file is written to the new location to which the path refers. Whenever `pbvi` is run from Privilege Management for Unix and Linux, the arguments should be checked to ensure that the user cannot change the path and no security hole is introduced.

Syntax

```
pbvi fullpathname
```

Arguments

Argument	Description
<code>fullpathname</code>	File to edit.

Files

None



Example: To display the contents of the file called `fullpathname` for editing, enter the following:

```
pbvi fullpathname
```



For more information, please see "`pbrun`" on page 465.

pblogarchive

- **Version 8.5 and earlier:** **pblogarchive** not available.
- **Version 9.0 and later:** **pblogarchive** available.

pblogarchive is the log archiving utility that can archive I/O logs and event logs from the original logserver onto an archive host.

It is installed on hosts configured as log servers and policy servers. It must be called from a host containing the event logs or I/O logs to be archived.

You must be logged in as root to use **pblogarchive**.

Syntax

```
pblogarchive [options]
  -e, --eventlog [filepathname]
  -E, --eventlogbase "<shell pattern>"
  -i, --iolog <filepathname>
  -I, --iologbase "<shell pattern>"
  -s, --serverinfo <archivehost>
  -l, --list[e|i]
pblogarchive -v | --version
pblogarchive -h | --help
```

Arguments

-e	Archive the event log specified in the settings file. Cannot be combined with the I/O log options (-i , -I).
--eventlog [=filepathname]	Archive an event log. Provide the absolute file path name of the event log to archive. If file path name is omitted, pblogarchive determines the event log from the settings file. Cannot be combined with the I/O log options (-i , -I).
-E, --eventlogbase "shell_pattern"	Archive event log(s) matching the shell pattern. Shell pattern must be enclosed in quotes. Cannot be combined with the I/O log options (-i , -I).
-i, --iolog <filepathname>	Archive the specified I/O log. Cannot be combined with the event log options (-e , -E).
-I, --iologbase "shell_pattern"	Archive I/O log(s) matching the shell pattern. Shell pattern must be enclosed in quotes. Cannot be combined with the event log options (-e , -E).
-s, --serverinfo <archivehost>	Specify the name or IP address of the destination archive host. This overrides logarchivehost in pb.settings .
-l, --list[e i]	List logfile locations in the log tracking DB. -le : Lists only event log information.

-li: Lists only I/O log information.

For event logs, displays the name of the original logserver, event log creation datetime in UTC, event log archive datetime in UTC(when the event log was first rotated), original event log path, current host where logfile is located/archived, and the current path name of the logfile.

For I/O logs, displays the name of the original logserver, I/O log creation datetime in UTC, uniqueid associated with the I/O log, I/O logfile sequence number if partitioned, original I/O log path, current host where logfile is located/archived, and the current path name of the logfile.



Example: Archives the event log named by the **eventlog** setting in **pb.settings**:

```
pblogarchive -e
```



Example: Archives a specific event log:

```
pblogarchive --eventlog=/var/log/my90pb.eventlog_20150512_161106
```



Example: Archive multiple event logs using a filename shell pattern:

```
pblogarchive -E "/var/log/my90pb.eventlog_2015*"
```



Example: Archive a specific I/O log:

```
pblogarchive -i /var/log/iolog/my90iolog.7ZSa1y
```



Example: Archive multiple I/O logs using a filename shell pattern:

```
pblogarchive -I "/var/log/iolog/my90iolog.7*"
```



Example: Archive an event log to a specific archive host (override settings file):

```
pblogarchive -s archost21 --eventlog=/var/log/my90pb.eventlog_20150512_161106
```



Example: List event log location as recorded by the log tracking database:

```
#pblogarchive -le

List of Event Logs:

Orig_Logserver,Date_Created_UTC,Date_Archived_UTC,Orig_Path,Current_Host,Current_Path
dev-erolnd-01.unix.ca.com,2015-05-2802:49:21,-,/var/log/ABCac90pb.eventlog,dev-erolnd-01.unix.ca.com,/var/log/ABCac90pb.eventlog

dev-erolnd-01.unix.ca.com,2015-05-28 02:48:00,2015-05-28 02:48:00,/var/log/ABCac90pb.eventlog,dev-erolnd-01.unix.ca.com,/var/log/ABCac90pb.eventlog_20150527_194800

dev-erolnd-01.unix.ca.com,2015-05-28 02:43:01,2015-05-28 02:44:34,/var/log/ABCac90pb.eventlog,dev-erolnd-01.unix.ca.com,/opt/ARCHIVELOGS3/eventlog/dev-erolnd-01.unix.ca.com/ABCac90pb.eventlog_20150527_194433
```



Example: List I/O log location as recorded by the log tracking database

```
# pblogarchive -li

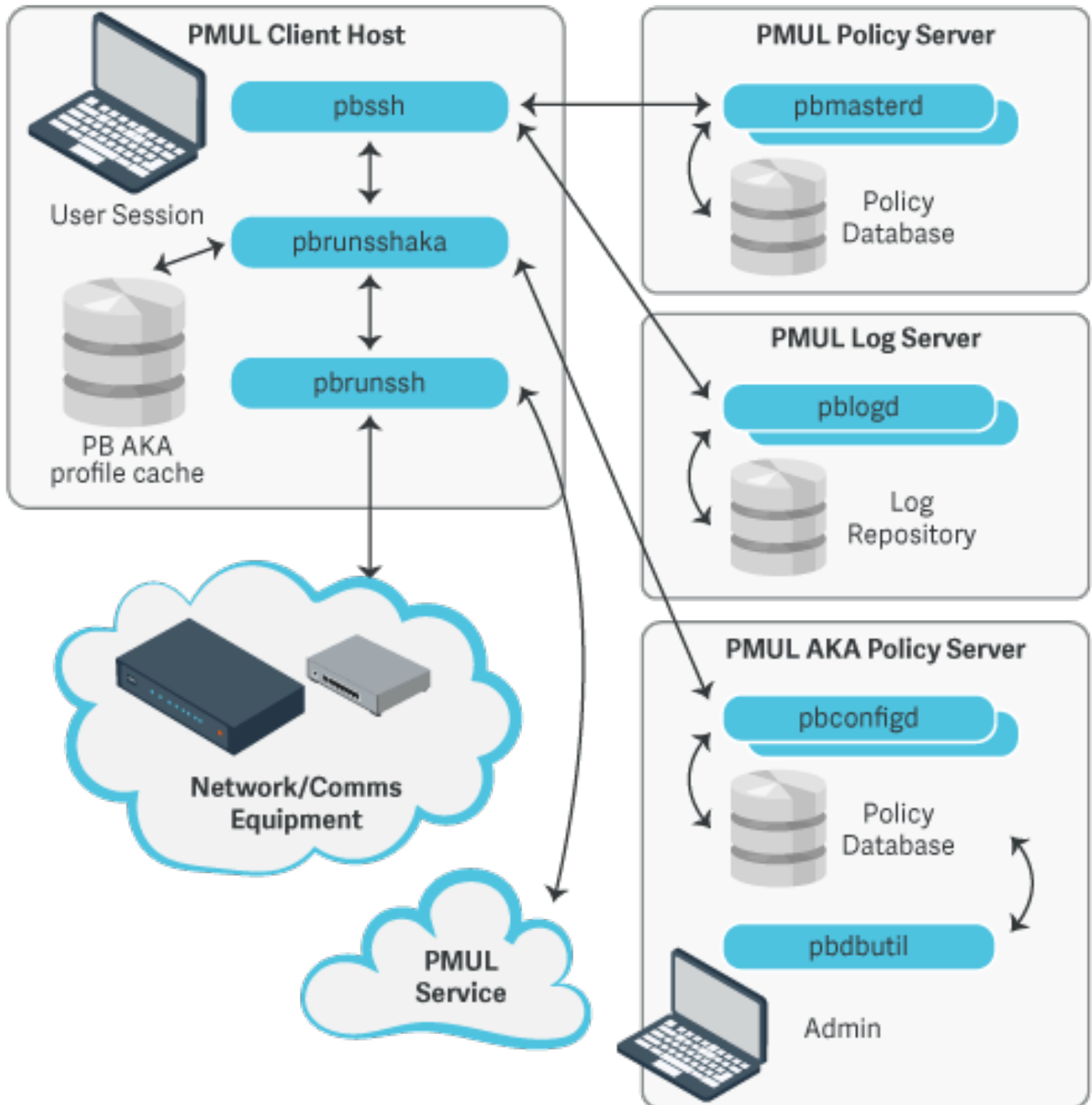
List of IO Logs:

Orig_Logserver,Date_Created_UTC,Unique_Id,Partial_Id,Orig_Path,Current_Host,Current_Path
dev-erolnd-01.unix.ca.com,2018-09-18 18:25:53,ac1420205ba1435717DE,1,/var/log/pbsudo/dev-eolog-08.unix.ca.com/pbsudo.iolog.kAWJLT,dev-erolnd-02.unix.ca.com,/opt/iolog/dev-eolog-08.unix.ca.com/testuser/20180918/pbsudo.iolog.kAWJLT.ac1420205ba1435717DE
```

Advanced Keystroke Action

Advanced Keystroke Action was introduced to allow control and audit of command line based network appliances, and was implemented as an enhancement to the **pbssh** feature. Full session logging provides a complete command audit trail through the existing session logging technology.

Advanced Keystroke Action differs from previous features in that instead of trying to apply command control as the user types, it emulates an interactive command line, and only then authorizes the command once the user has pressed **Enter** to execute the command. This means the policy can try to match the command it has received in context to the task the user is performing, and it can choose to rewrite the command, accept it, or reject it. It also allows the policy to change the user environment as they carry out their tasks, for example, changing prompts or tab completion.



Get Started with Advanced Keystroke Action

Configure Privilege Management for Unix and Linux Policy

Advanced Keystroke Action is implemented on top of existing **pbssh** technology and appropriate configuration is required in the policy script file. For example:

```
if(pbclientname=="pbssh") {
myadmins={"admin1"};
myswitches={"cisco1"};
if(pbclientmode=="pbssh") {
if(search(myadmins,user) >= 0 && search(myswitches,host) >= 0) {
accept;
} else {
reject;
}
} else if(pbclientmode=="run") {
args=split(argv[argc-1], " ");
argslen = length(args);
if(argslen > 0 && search(myswitches,args[argslen-1]) >= 0) {
keystrokeactionprofile="cisco profile1";
accept;
} else {
reject;
}
}
}
```

Configure Password Safe

Configure Password Safe on the Advanced Keystroke Action primary server or Policy Servers.

i To provide automatic authentication, Password Safe can be configured to retrieve logon credentials. For information on this configuration, including the specification of **pkrunfile**, **pk_cert**, and **pk_servers** in the **pb.settings** configuration file, please see "**pbssh**" on page 468

Configure the Advanced Keystroke Action Primary Server or Policy Servers

If the Advanced Keystroke Action is being configured for the first time, the policy database will need to be created on the primary AKA server or primary Policy Server.

1. Specify the Advanced Keystroke Action Policy Database by adding the keyword **advkeystrokeactionpolicydb** to the **pb.settings** on all server hosts.
2. If you are using Registry Name Service you will need to define your Primary Server in the RNS using:


```
pbadmin --svc -u '{"svcname":"dflt_akapolicy_service",
  "svc":"advkeystrokeactionpolicy",
  "cn": "<policy server fqdn>",
  "role":"primary"}
```

3. Create the policy database (which includes a demo policy that can be used as a basis for further configuration and learning)

```
pbadmin --aka -n
```

The **pbadmin** command can then be used to export and re-import Advanced Keystroke Action policies. The Privilege Management for Unix and Linux script policy keyword must match a valid Advanced Keystroke Action policy name for the session to run.

Maintenance and Configuration of Advanced Keystroke Action Policy

pbadmin has been enhanced to provide methods to maintain the Advanced Keystroke Action Policy on the Primary Policy server. This includes a new section (**--aka**) with new options.

Usage

```
pbdbutil --aka [<options>] [ <file> <file> ...]
```

Options for AKA client:

-R	Force refresh of the client aka profile cache file(s)
args	Optional. The arguments to the function.

Options for Advanced Keystroke Action profile server database management:

-n	Create new AKA database
-l	List all AKA configurations in database
-s <[- +]attribute>	Sort the list of records by attribute name (asc/desc)
-i <file>	Import AKA configuration file
-e <name> <file>	Export specified AKA configuration
-g <name>	Get AKA configuration by name
-d <name>	Delete AKA configuration
-u { "name" : "<name>", "cfg": { json param... } }	Update AKA configuration

Advanced Keystroke Action Policy

Policies for Advanced Keystroke Action are different from normal Privilege Management for Unix and Linux policies because they have a different function to fulfill. The policy is defined as a JSON object split into a number of separate sections including variables, prerun, complete actions, macros, readonly, and policy. The primary section of the policy is an ordered list of **match** and **action** nodes. These match the command line, or other variable, and perform specified actions if the match is successful.

Preamble

The preamble defines the name of the configuration and whether debug can be enabled.



Example: Through the command line:

```
set akadebug 10
```

```
{
  "name": "demo",
  "cfg": {
    "debug_enabled": true,
    ...
  }
}
```

Variables

The variables section allows the configuration of variables that can be used within the policy. These variables, by default are read/write within the policy but may be defined as **readonly** in the readonly section. All JSON data types are supported, including booleans (true/false), strings, integers, floating point numbers (reals), and arrays of datatypes using the JSON format. These can then be referred to by name using a notation similar to bash. For example, **`\${varname}`** or **`\${arrayelement[2]}`**. Some of the variables are special in that they can only hold specific values. For example, **editor** can only be **emacs** or **vi**, and **keymap** can only hold specific definitions of keys (similar to the NetBSD libedit or GNU readline names).

The variable **input** is the current command line, and if it is rewritten it will change the command line. If it is set to **null**, it removes the current input and returns the user to the command line.



Example:

```
"variables": {
  "editor": "emacs", "prompt": "$ ", "ro_var": "this variable is readonly", "remote_
prompt":
  "##prompt##:",
  "keymap": [{ "key": "^I", "value": "ed-complete" }
]
},
```

Key definitions by default are listed below.

In **vi** input mode, input characters are bound to the following editor commands by default:

Ctrl-D, EOF	vi-list-or-eof
Ctrl-H, BS	vi-delete-prev-char
Ctrl-J, LF	ed-newline
Ctrl-M, CR	ed-newline
Ctrl-Q	ed-tty-start-output
Ctrl-S	ed-tty-stop-output
Ctrl-U	vi-kill-line-prev
Ctrl-V	ed-quoted-insert
Ctrl-W	ed-delete-prev-word
Ctrl-[, ESC	vi-command-mode
Ctrl-\\, QUIT	ed-tty-sigquit
Ctrl-?, DEL	vi-delete-prev-char

All other input characters except the NUL character (**Ctrl-@**) are bound to **ed-insert**.

In **vi** command mode, input characters are bound to the following editor commands by default:

Ctrl-A	ed-move-to-beg
Ctrl-C, INT	ed-tty-sigint
Ctrl-E	ed-move-to-end
Ctrl-H, BS	ed-delete-prev-char
Ctrl-J, LF	ed-newline
Ctrl-K	ed-kill-line
Ctrl-L, FF	ed-clear-screen
Ctrl-M, CR	ed-newline
Ctrl-N	ed-next-history
Ctrl-O	ed-tty-flush-output
Ctrl-P	ed-prev-history
Ctrl-Q	ed-tty-start-output
Ctrl-R	ed-redisplay
Ctrl-S	ed-tty-stop-output
Ctrl-U	vi-kill-line-prev

Ctrl-W	ed-delete-prev-word
Ctrl-[, ESC	em-meta-next
Ctrl-\, QUIT	ed-tty-sigquit
Space	ed-next-char
#	vi-comment-out
\$	ed-move-to-end
%	vi-match
+	ed-next-history
,	vi-repeat-prev-char
-	ed-prev-history
.	vi-redo
/	vi-search-prev
0	vi-zero
1 to 9	ed-argument-digit
:	ed-command
;	vi-repeat-next-char
?	vi-search-next
@	vi-alias
A	vi-add-at-eol
B	vi-prev-big-word
C	vi-change-to-eol
D	ed-kill-line
E	vi-end-big-word
F	vi-prev-char
G	vi-to-history-line
I	vi-insert-at-bol
J	ed-search-next-history
K	ed-search-prev-history
N	vi-repeat-search-prev

O	ed-sequence-lead-in
P	vi-paste-prev
R	vi-replace-mode
S	vi-substitute-line
T	vi-to-prev-char
U	vi-undo-line
W	vi-next-big-word
X	ed-delete-prev-char
Y	vi-yank-end
[ed-sequence-lead-in
^	ed-move-to-beg
_	vi-history-word
a	vi-add
b	vi-prev-word
c	vi-change-meta
d	vi-delete-meta
e	vi-end-word
f	vi-next-char
h	ed-prev-char
i	vi-insert
j	ed-next-history
k	ed-prev-history
l	ed-next-char
n	vi-repeat-search-next
p	vi-paste-next
r	vi-replace-char
s	vi-substitute-char
t	vi-to-next-char
u	vi-undo

v	vi-histedit
w	vi-next-word
x	ed-delete-next-char
y	vi-yank
	vi-to-column
~	vi-change-case
Ctrl-?, DEL	ed-delete-prev-char
Meta-O	ed-sequence-lead-in
Meta-[ed-sequence-lead-in

In **emacs** mode, input characters are bound to the following editor commands by default:

0 to 9	ed-digit
Ctrl-@, NUL	em-set-mark
Ctrl-A	ed-move-to-beg
Ctrl-B	ed-prev-char
Ctrl-C, INT	ed-tty-sigint
Ctrl-D, EOF	em-delete-or-list
Ctrl-E	ed-move-to-end
Ctrl-F	ed-next-char
Ctrl-H, BS	em-delete-prev-char
Ctrl-J, LF	ed-newline
Ctrl-K	ed-kill-line
Ctrl-L, FF	ed-clear-screen
Ctrl-M, CR	ed-newline
Ctrl-N	ed-next-history
Ctrl-O	ed-tty-flush-output
Ctrl-P	ed-prev-history
Ctrl-Q	ed-tty-start-output
Ctrl-R	ed-redisplay
Ctrl-S	ed-tty-stop-output

Ctrl-T	ed-transpose-chars
Ctrl-U	ed-kill-line
Ctrl-V	ed-quoted-insert
Ctrl-W	em-kill-region
Ctrl-X	ed-sequence-lead-in
Ctrl-Y	em-yank
Ctrl-Z, TSTP	ed-tty-sigtstp
Ctrl-[, ESC	em-meta-next
Ctrl-\, QUIT	ed-tty-sigquit
Ctrl-]	ed-tty-dsusp
Ctrl-?, DEL	em-delete-prev-char
Ctrl-Meta-H	ed-delete-prev-word
Ctrl-Meta-L	ed-clear-screen
Ctrl-Meta-_	em-copy-prev-word
Meta-0 to 9	ed-argument-digit
Meta-B	ed-prev-word
Meta-C	em-capitol-case
Meta-D	em-delete-next-word
Meta-F	em-next-word
Meta-L	em-lower-case
Meta-N	ed-search-next-history
Meta-O	ed-sequence-lead-in
Meta-P	ed-search-prev-history
Meta-U	em-upper-case
Meta-W	em-copy-region
Meta-X	ed-command
Meta-[ed-sequence-lead-in
Meta-b	ed-prev-word
Meta-c	em-capitol-case

Meta-d	em-delete-next-word
Meta-f	em-next-word
Meta-l	em-lower-case
Meta-n	ed-search-next-history
Meta-p	ed-search-prev-history
Meta-u	em-upper-case
Meta-w	em-copy-region
Meta-x	ed-command
Ctrl-Meta-?	ed-delete-prev-word

The **prompt** variable is the prompt that the user will see to type at. The **remote_prompt** is important in that AKA attempts to wait for the remote prompt before letting the user type at the command line. This should make interaction clearer and more intuitive without mixing input and output characters in the input/output terminal data stream.

Other special variables are:

- **runuser:** The remote run user
- **runhost:** The fully qualified domain name of the the remote run host
- **runhostname:** The simple hostname of the remote run host
- **runcmd:** The command line passed to **pbssh**
- **argv:** The command line array passed to **pbssh**
- **prompt_wait:** The maximum time to wait for remote command to run (in milliseconds)
- **history:** The number of command line history entries to save for the next session
- **input:** The submitted command line
- **prerun_prompt_wait:** The maximum time to wait for remote command to run when executing the pre-run as the user session starts (in milliseconds)
- **prerun_slowdown:** The delay time added to each prerun command to slow down the prerun (in milliseconds)
- **remote_prompt:** The remote prompt, used by Advanced Keystroke Action profiles to synchronize the terminal session input and output sequences
- **akadebug:** Built-in debugging to allow the policy writer to diagnose issues in policy
- **pkcert:** The value of the **pk_cert** passed into the Advanced Keystroke Action service
- **pkserver:** The value of the **pk_server** passed into the Advanced Keystroke Action service
- **pkdomain:** The value of the **pk_domanit** passed into the Advanced Keystroke Action service



For more information, please see the *NetBSD editline(7)* man page.

Prerun

The prerun section is an array of string commands that are submitted to the remote system before any user commands can be submitted. It is recommended, if possible, to define prompts, etc, so they can be waited for using **remote_prompt** variable. The strings are submitted

one after another with a short delay, but with not intelligent processing. If processing is required is recommended that normal policy is used.

Example:

```
"prerun": [ "PS1=##prompt##:", "unset PROMPT_COMMAND" ],
```

Filter

The filter section is an array of strings that are filtered from the session output and are therefore not displayed on the client terminal session.

Example:

```
"filter": [ "Filtered string one", "Filtered string two" ],
```

Login_sequence

The login sequence object details what happens with a successful or unsuccessful login. It details whether the user is allowed any input during the login phase, the successful/unsuccessful messages and strings that are matched to determine whether it was successful:

Example:

```
"login_sequence" : {
  "input_allowed" : true,
  "successful" : "*** Login sequence successful ***",
  "denied" : "*** Failed login sequence - access denied ***",
  "prematch" : [
    {"Last login: " : true },
    {"Access denied" : false }
  ]
},
```

Completion

Completion is a list of strings that can be used to complete command lines (usually when the **ed-complete** key is submitted). If more than one string can complete the command, the list will be provided. If only one matches, the current command line is rewritten to the completion string. There are two types of completion value. A line completion, where the who line will be compared up to the cursor, and word completion that simply checks the previous word.

**Example:**

```
"complete": [ { "value": "compabcd", "type": "line" }, {"value": "compadef", "type": "line"}, {"value": "compxyz", "type": "word" } ],
```

Macros

Macros are an ordered list of regular expression patterns and rewrite strings. It is used to simply rewrite the current command line if patterns match when return is pressed. This should simplify the majority of policy that are simple rewrites.

**Example:**

```
"macros": [
  { "pattern": "macroa(b)(c)d", "value": "echo macro rewrites to ${1} ${2}" }
],
```

When the user types **macroabcd**, it rewrites the command line to **echo macro rewrites to b c**, and it executes the command on the remote system.

Readonly

The readonly section is a simple list of variables that are readonly and cannot be rewritten. This is useful for constant strings or messages.

**Example:**

```
"readonly": {
  "ro_var": true
},
```

Policy

The policy section is an ordered list of matching conditions and actions that are taken if the match is positive. Matching is processed against variables or user input, and can be any normal operation, including regular expressions. Actions can be:

- **Accept:** Immediately accept the current command line and submit the command line to the remote host to execute.
- **Reject:** Immediately reject the current command line, optionally displaying a message.
- **Restart:** Restart processing the policy list from the beginning.
- **Readonly:** Mark the specified variable readonly for the rest of the session.
- **Disable:** Execute the current command line, but disable command line processing until either the specified string is matched in the output, or the remote prompt is once again displayed. This allows interactive sessions to be authorized.
- **Unset:** Unset the specified variable.
- **Printf:** Display the specified message to the users terminal session, processing specified arguments to make up the message.

- **Sendf:** Send the specified message to the remote application as if typed, processing specified arguments to make up the input.
- **Set:** Set the specified variable to a constant or calculated value. Constant values can include any data type, or can reference another variable. Calculated values can be integer or floating point math, shift or logic values, substrings, substituted strings, upper/lower case strings, split or sprint strings. Arguments can be constants or specified variable values, or the arguments matched from the regular expression match.
 - Operators on integers include add(+), subtract (-), divide (/), multiply (*), logical AND (&&), bitwise AND (&), logical OR (||), bitwise OR (|), NOT (!), shift left (<<), shift right (>>), modulo divide (%).
 - Operators on floating point include add(+), subtract (-), divide (/), multiply (*).
 - Operators on strings include concatenation (+), substitute (**sub**), substring (**substr**), uppercase (**upper**), lowercase (**lower**), split/tokensize string (**split**) and sprintf to format a string with arguments.
 - Operators on arrays include append/extend (+), remove (-) and index set (=).

The match node consists of a singular expression match, or an array of called **and** or **or** which are evaluated to a logical AND or OR of the constituent singular expression matches.



Example: Simple match:

```
"match": { "op": "=", "${var}": "var1", "value": 123 }
```



Example: Logical AND of simple matches:

```
"match": { "and" = [
  { "op": "=", "var": "${var1}", "value": 123 },
  { "op": "=", "var": "${var2}", "value": 678 }
]}
```



Example: Logical OR of simple matches:

```
"match": { "or" = [
  { "op": "=", "var": "${var1}", "value": 123 },
  { "op": "=", "var": "${var2}", "value": 678 }
]}
```

A Simple Example of Policy



Example: This shows a simple policy where the command line is matched against the user typing **PS1=.....** If it matches, the command is rejected with a suitable message to the user.

```
"policy": [
  {
    "match": {
```



```
"op": "=~",  
"var": "${input}",  
"value": "PS1=.*"  
},  
"actions": [  
  {  
    "action": "reject",  
    "value": "sorry, you cannot reset the prompt"  
  }  
]  
}, ...
```



For more examples, please see the demo policy in the default Advanced Keystroke Action database.

Advanced Keystroke Action Logging / Events

Similarly to Privilege Management for Unix and Linux session logging, Advanced Keystroke Action sessions can be logged and events collated. The keyword **advkeystrokeactionevents yes** should be added to **pb.settings**. By default, the events will be collated in the event database or they can be logged in CSV or JSON to syslog, a normal text file, or sent to a program using the keyword **eventdestinations**.



Example: An Advanced Keystroke Action log:

```
{ "hostname": "pbuild", "evtname": "aka_start_session", "service": "pbrunsshaka9.4.4-03_
debug", "who": "root", "severity": 16, "utc": "2017-06-23
11:19:00", "progname": "pbrunsshaka9.4.4-03_debug", "version": "9.4.4-03_
debug", "arch": "linux.x86-64", "data":
{ "runhost": "cisco1", "hostname": "pbuild", "pid": 23319, "runuser": "admin", "sid": 23319, "uid":
0 } }
```

```
{ "hostname": "pbuild", "evtname": "aka_accept_cmd", "service": "pbrunsshaka9.4.4-03_
debug", "who": "root", "severity": 16, "utc": "2017-06-23
11:19:09", "progname": "pbrunsshaka9.4.4-03_debug", "version": "9.4.4-03_
debug", "arch": "linux.x86-64", "data": { "hostname": "pbuild", "pid": 23319, "cmd": "show ip
interface brief", "sid": 23319, "uid": 0 } }
```

```
{ "hostname": "pbuild", "evtname": "aka_reject_cmd", "service": "pbrunsshaka9.4.4-03_
debug", "who": "root", "severity": 16, "utc": "2017-06-23
11:19:11", "progname": "pbrunsshaka9.4.4-03_debug", "version": "9.4.4-03_
debug", "arch": "linux.x86-64", "data":
{ "hostname": "pbuild", "pid": 23319, "cmd": "wexit", "policy":
{ "action": "reject", "sid": 23319, "uid": 0 } }
```

```
{ "hostname": "pbuild", "evtname": "aka_terminate", "service": "pbrunsshaka9.4.4-03_
debug", "who": "root", "severity": 16, "utc": "2017-06-23
11:19:11", "progname": "pbrunsshaka9.4.4-03_debug", "version": "9.4.4-03_
debug", "arch": "linux.x86-64", "data":
{ "hostname": "pbuild", "pid": 23319, "cmd": "exit", "policy":
{ "action": "terminate", "sid": 23319, "uid": 0 } }
```

```
{ "hostname": "pbuild", "evtname": "aka_end_session", "service": "pbrunsshaka9.4.4-03_
debug", "who": "root", "severity": 16, "utc": "2017-06-23
11:19:11", "progname": "pbrunsshaka9.4.4-03_debug", "version": "9.4.4-03_
debug", "arch": "linux.x86-64", "data": { "status":
{ "status": 0, "hostname": "pbuild", "pid": 23319, "sid": 23319, "uid": 0 } }
```



For more information, please see *"Auditing and Logging"* on page 114.

File Integrity Monitoring

Privilege Management for Unix and Linux includes File Integrity Monitoring (FIM) to enhance system security and audit. FIM policies can be configured to schedule regular checks of the integrity of operating systems, software applications, and customer data. This verifies file permissions, ownership, and even cryptographic checksums, producing details reports for security alerts, vulnerability assessments, and audit.

Overview

FIM policies are configured and maintained in a centralized repository. FIM clients are assigned to specific policy, and automatically retrieve and use these policies to compare the local file system against a system baseline. Any policy violations or inappropriate changes to the file system are detailed in a report which is compiled and sent back to the central repository for future reference. Events are also generated to alert administrators of the security transgression.

Policy Configuration

FIM Policy configuration is stored as a JSON script. Each policy is named and details which file systems and files to verify, what aspects of each file to check, and assigns a perceived risk level to any violation. Each individual client host is assigned to a single named policy (or uses the default policy if unassigned). Each policy is named and is subject to change control if it is enabled.

Each policy is split into sections dealing with predefinitions of what aspects of individual files and file types to check, a list of included paths and files and what predefinitions to use for them, and a list of paths or files to exclude from the file integrity check.

Predefinitions Section

This section of the policy defines what checks need to be made on specific file types.



Example:

```
"sysconf": {
  "exec": { "ino": true, "mode": true, "uid": true, "gid": true, "pmask": "022", "size":
true, "mtime": true, "ctime": true, "hash": true, "risk": 10 },
  "script": { "ino": true, "mode": true, "uid": true, "gid": true, "pmask": "022", "size":
true, "mtime": true, "ctime": true, "hash": true, "risk": 10 },
  "dev": { "uid": true, "gid": true, "mode": true, "risk": 10 },
  "other": { "ino": true, "mode": true, "uid": true, "gid": true, "pmask": "002", "size":
true, "mtime": true, "ctime": true, "risk": 6 }
}
```

The above predefinition details file system attributes to check for executable files, scripts, devices, and then all other files.

File types are defined as:

- **dir:** Directory
- **dev:** Character or block device
- **link:** Symbolic link
- **sock:** File socket
- **script:** Script (! interpreter)

- **exec:** Executable file
- **file:** Any other file type

A file type of **all** overrides all other entries within that predef.

A file type of **other** matches anything that isn't matched by more specific types.

The complete list of file system attribute checks are:

dev	Boolean	Check the file device
ino	Boolean	Check the file inode
pdev	Boolean	Check the parent device
pino	Boolean	Check the parent inode
mode	Boolean	Check the file mode (07777, includes owner/group/other and setuid/setgid)
uid	Boolean	Check the uid
gid	Boolean	Check the gid
size	Boolean	Check the file size
hash	Boolean	Check the SHA256 hash of the file
hash_smaller	Integer	Check the SHA256 hash if the file is smaller than <num> kilobytes
size_min	Boolean	Check the file is the same or larger (for log files)
mtime	Boolean	Check the modification time
mtime_later	Boolean	Check the modification is equal or later (for log files)
ctime	Boolean	Check the change time
ctime_later	Boolean	Check the change time is equal or later (for log files)
perm	07777	Check specified permissions are set
pmask	07777	Check specified permissions are NOT set
risk	Integer	Arbitrary number, Magnitude denotes risk
own	Integer/String	UID or Username of ownership of the file
gown	Integer/String	GID or Group ownership of the file
file	Boolean	Check the file's file name
ftype	Boolean	Check the file's type



Note: When symbolic links are scanned and checked, and the link is not broken, the link target's canonicalized name is reported and the link target's device and inode. These are reported as fields **linktarget**, **linkdev**, and **linkino**. This data is stored and checked within the hash field.



For more information on **perm** and **pmask**, please see **man umask**.

Include Section

The include section lists all those files to include in the file system scan. Each entry defines a file specification which may be wildcarded, the name of the predef check that is performed, whether to recurse down the folder (if one is specified), whether to follow links into different directories, and whether to cross devices.

When wildcards are used on directory names, any dot files in that directory are not matched. To match files whose name begins with a dot, the path and directory name must not include wildcards.

Files are scanned in order of entries in the include section, according to the first pattern that matches the file. Any subsequent file matching patterns is ignored.



Example: If we generally want **/etc/*** scanned with the **sysconf** predef, yet we specifically want **/etc/passwd** scanned with the **log** predef, the rule for **/etc/passwd** must appear before the entry for **/etc**.

```
"include": [
  { "path": "/etc/mtab",      "chk": "log",      "recurse": false },
  { "path": "/etc/motd",    "chk": "log",      "recurse": false },
  { "path": "/etc/passwd",  "chk": "log",      "recurse": false },
  { "path": "/etc/shadow",  "chk": "log",      "recurse": false },
  { "path": "/etc/*",       "chk": "sysconf",  "recurse": true, "follow": false, "xdev": true
  },
  { "path": "/proc",        "chk": "log",      "recurse": false },
  { "path": "/mnt",         "chk": "log",      "recurse": false },
  { "path": "/boot/*",     "chk": "sysconf",  "recurse": true, "follow": false, "xdev": true
  },
  { "path": "/bin/*",       "chk": "bin",      "recurse": true, "follow": false, "xdev": true
  },
  { "path": "/var/log/*",   "chk": "log",      "recurse": true, "follow": false, "xdev": true
  },
  { "path": "/var/adm/*",   "chk": "log",      "recurse": true, "follow": false, "xdev": true
  }
]
```

Exclude Section

The exclude section defines a list of wildcard strings that excludes files that have previously been specifically included in the previous section.


Example:

```
\exclude': [
  '\root/.sh_history',
  '\home/*',
  '\etc/pb.db'
]
```

Policy Configuration and Maintenance

All configuration and maintenance of FIM policy is carried out using the **pbdbutil** utility. The utility has been enhanced to provide methods to retrieve, store, and list policies, as well as retrieve and summarize the reports generated from client hosts. The utility is also used on client hosts to initiate the file system checks, either from the command line, or scheduled using **cron**.

Syntax

Usage

```
pbdbutil --fim [<options>] [ <file> <file> ...]
```

Options for FIM client:

-r	Run FIM check
-U	Run FIM check and update database

Options for client remote management:

--client='{ params }'	Specify client REST parameters (NB use of = for optional parameter)
------------------------------	---

Options for FIM server database management:

-n	Create new FIM database
-l	List all FIM configurations in database
-l	Add an extra '-l' to list host assignments
-s <[- +]attribute>	Sort the list of records by attribute name (asc/desc)
-i <file>	Import FIM configuration file
-e <name> <file>	Export specified FIM configuration
-g <name>	Get FIM configuration by name
-d <name>	Delete FIM configuration

<code>-d { "cfg" : { "name" : "<wildcard>" } }</code>	Delete FIM configuration matching wildcard
<code>-u { "name" : "<name>", "cfg": { json param... } }</code>	Update FIM configuration
<code>-A <name> <host(s)></code>	Assign host to configuration
<code>-X <host(s)></code>	Assign host to configuration
<code>-g { "rpt" : { "uuid" : "<uuid>" } }</code>	Get FIM report
<code>-g { "rpt" : { params ... } }</code>	Retrieve report summarized from multiple reports
<code>--format '["header", "header2", ...]'</code>	Define retrieved fields when using CSV report
<code>-d { "rpt" : { "uuid" : "<uuid>" } }</code>	Delete FIM report
<code>-d { "rpt" : { params ... } }</code>	Delete FIM report(s)
<code>-L {</code>	Retrieve, List or Delete FIM reports, with the attributes: <ul style="list-style-type: none"> • <code>["name" : "<wildcard>"]</code>, • <code>["uuid" : "<uuid>"]</code>, • <code>["host" : "<wildcard>"]</code>, • <code>["older" : <epoch>, ["newer" : <epoch>]</code>, • <code>["older" : { "years" : "months" : n, "days" : "hours" : n }</code> • <code>["newer" : { "years" : n, "months" : n, "days" : n, "hours" : n }</code> • <code>["updates" : <bool>]</code>, • <code>["risk" : <lvl>]</code>, • <code>["risk_higher" : <lvl>]</code>, • <code>["risk_lower" : <lvl>]</code>, • <code>["regexp" : true]</code>
<code>-s <[-+]attribute></code>	Sort the list of records by attribute name (asc/desc)

Initial Setup and Configuration of the Service

1. Specify the FIM database by adding the keyword **fileintegritydb** to the **pb.settings** on all hosts.
2. If you are using Registry Name Service, define your primary server in the RNS database using:

```
pbdbutil --svc -u '{ "svcgname":"dflt_fim_service", "cn" : "host", "role", "primary" }'
```

If you are not using Registry Name Service the **submitmasters** keyword is used to specify the FIM Server.

3. Initially, on the centralized server, the FIM database needs to be created using:

```
pbdbutil --fim -n
```

4. A configuration must be defined and stored using:

```
pbdbutil --fim -u '{ "cfg": { ... } }'
```

5. Once this has been performed, each client is assigned a configuration using:

```
pbdbutil --fim -A <cfgname> <hostname>
```

6. On the client, run the following command to setup the initial baseline database and check the results are expected.

```
pbdbutil --fim -U
```

7. Once this has been completed, add a **cron** job to run the check on a regular basis:

```
0 0 * * * /usr/sbin/pbdbutil --fim -r > /dev/null 2>&1
```

Each run uploads the results to the centralized report repository. If configured by adding the setting **fileintegrityevents** on the primary service, each run also sends an event to the **eventdb** database.

File Integrity Monitor Reports

Reports consist of a JSON array of results, containing **before** (for example, existing entries in the baseline database) and **after** or new items which represent the current file system file status. Depending upon the checks run, various attributes are reported in each **before** or **after** item.

All character data (for example, file names) must consist of valid UTF-8 characters. Character bytes that are not valid UTF-8 characters are reported as **[<hex value of the byte>]**, and further, the slash is escaped and appears as two slashes.

For example, **\\[98]** represents the byte value **98** hex, or **10011000** binary.



Example:

```
# pbdbutil -P --fim -r
{
  "rpt": [
    {
      "path": "/etc",
      "risk": 6,
      "change": "updated",
      "before": {"size": 249, "mtime": "2016-09-23 09:13:19", "ctime": "2016-09-23
09:13:19", "name": "resolv.conf",
      "dev": 2051, "ino": 3415679},
      "after": {"size": 0, "mtime": "2016-09-26 08:20:09", "ctime": "2016-09-26
08:20:09", "name": "foo", "dev": 2051, "ino": 3415679 }
    },
    {
      "path": "/var/log",
      "risk": 4,
```



```

        "change": "new",
        "after": {"pdev": 2051,"pino": 3933733, "mode": "600", "uid": 0, "gid": 0,
"size": 1493, "mtime": "2016-09-26 08:20:01",
        "ctime": "2016-09-26 08:20:01", "ftype": "file", "name": "cron", "hash":
null, "dev": 2051, "ino": 3933604 }
    }
    ],
    "new": 1,
    "deleted": 0,
    "updated": 1,
    "policy": 0,
    "total": 2,
    "max_risk": 6,
    "name": "default",
    "hostname": "pbuild",
    "uuid": "e62df7ea-980c-4252-a924-79a71a6ccc0f",
    "whoami": "ctaylor"
}
    
```

Details at the end of the report include the number of new files, deleted files, updated files and those reported because they do not meet policy. Also detailed at the end of the report is the maximum risk (for example, the highest risk item that was tagged as updated), the name of the configuration, the hostname, a unique uuid of the report, who ran it, and the number of updates that were logged. The attributes name, inode, device, and path are always logged for every item so offending files can be uniquely identified on the file system.

As well as the standard report, the customer can choose to report in CSV using the **-C** option on the command line.



Example:

```

rpt{*}.path,max_risk,name,host,uuid,rundate,total,new,updated,policy,deleted,rpt
{*}.risk,rpt{*}.change,rpt{*}.before.name,rpt{*}.before.ftype,rpt{*}.before.dev,rpt
{*}.before.ino,rpt{*}.before.pdev,rpt{*}.before.pino,rpt{*}.before.size,rpt
{*}.before.ctime,rpt{*}.before.mtime,rpt{*}.before.mode,rpt{*}.before.gid,rpt
{*}.before.uid,rpt{*}.before.hash,rpt{*}.after.name,rpt{*}.after.ftype,rpt
{*}.after.dev,rpt{*}.after.ino,rpt{*}.after.pdev,rpt{*}.after.pino,rpt{*}.after.size,rpt
{*}.after.ctime,rpt{*}.after.mtime,rpt{*}.after.mode,rpt{*}.after.gid,rpt
{*}.after.uid,rpt{*}.after.hash

/etc/pki,6,default,-,eff6b3c4-5a2e-4d10-8352-
3c7e1a1d4523,-,10,7,3,0,0,6,updated,nssdb,-,2051,3408221,-,-,-,2016-09-22 15:02:45,2016-
09-22 15:02:45,-,-,-,-,nssdb,-,2051,3408221,-,-,-,2016-09-26 08:09:35,2016-09-26
08:09:35,-,-,-,-

/etc,6,default,-,eff6b3c4-5a2e-4d10-8352-
3c7e1a1d4523,-,10,7,3,0,0,6,new,-,-,-,-,-,-,-,-,-,-,resolv.conf,file,2051,3428436,2
051,3407873,249,2016-09-26 08:19:15,2016-09-26 08:19:15,644,0,0,-
    
```

The summary information is added to every row, thereby *flattening* the CSV report. Additionally, a list of headers can be provided using the **--format** option.

**Example:**

```
pdbbutil -C -fim -r --format='["rpt{*}.path","max_risk","name","host","uuid","rpt{*}.change"]'
```

REST API

Introduction

A REST API is available to allow Privilege Management data to be configured, customized, and retrieved by other software. The API is web-based and uses industry standard modern components, connectors, and data elements within a distributed and secure enterprise environment.

The REST API is bundled with the main installation and configured in **pbinstall**:

```
15  Install REST Services?          [yes]
72  REST Service installation directory?  [/usr/lib/beyondtrust/pb/rest]
```

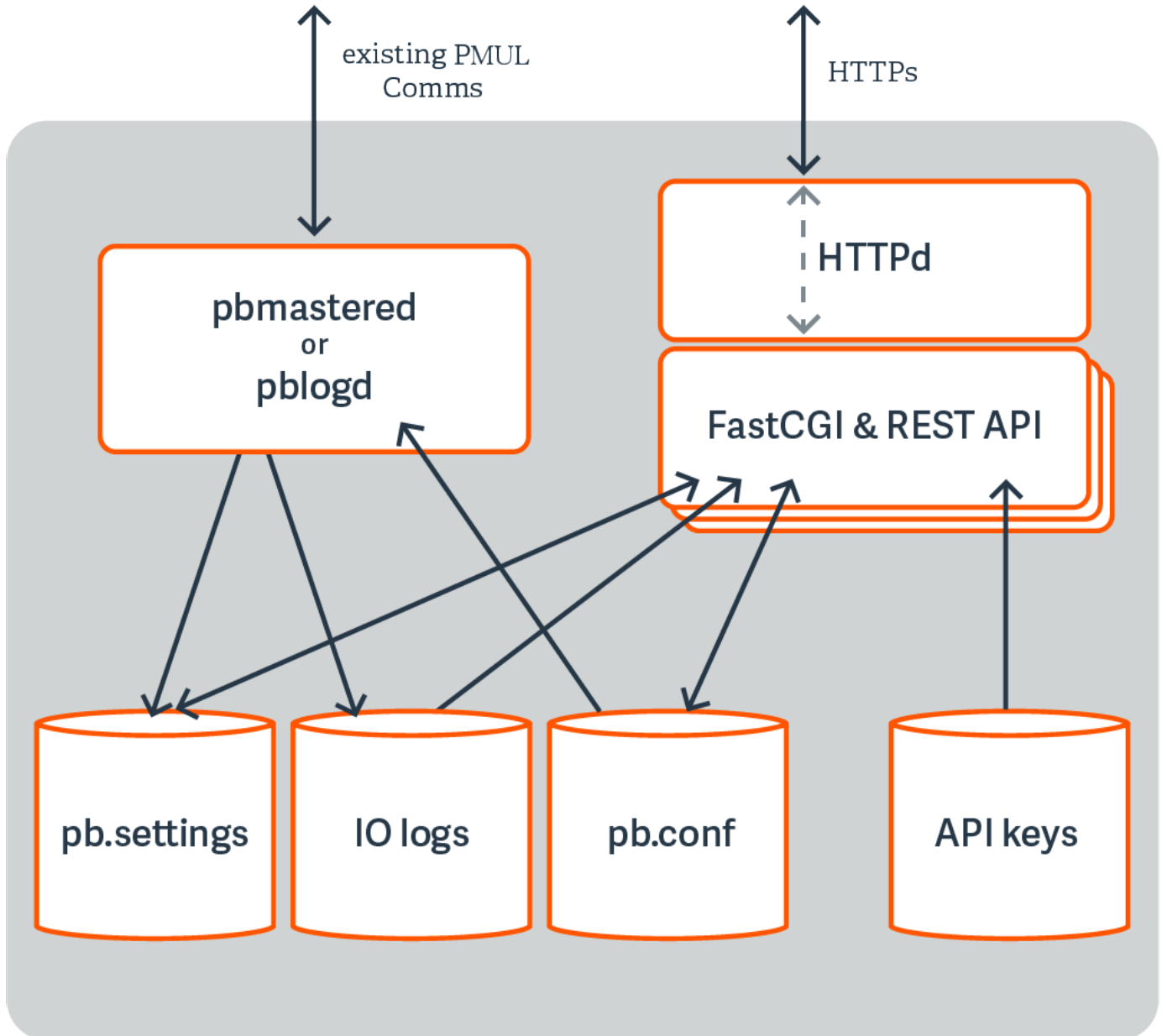
The menu response shown above installs the REST API web server components. When installing any server components, REST API installation becomes mandatory.

```
73  Install REST API sample code?  [yes]
74  REST API sample code directory?  [/usr/local/lib/pbrest]
```

The menu response shown above installs the Java and scripting example code.

The API can be used with Privilege Management for Unix and Linux v7.1.0 and later.

Architecture Overview



Functionality

Settings

- **Get All Settings:** Gets all of the settings from **pb.settings** (or equivalent). Each setting has one of 4 distinct types:
 - String
 - Boolean
 - List of Strings
 - **altsubmitmasters:** Has a special list of Privilege Management for Unix and Linux objects
- **Get Individual Settings:** Gets an individual setting as specified on the URL.
- **Put Setting:** Puts (modify) a setting into the **pb.settings** file. The type must correspond to the original setting type.
- **Get Settings file as attachment:** Retrieves the whole **pb.settings** file as a binary attachment.

License

- **License Get:** Retrieves the text value of the license.

Policy

- **Policy List Dir:** Lists all of the files in a given directory (without checking if they are iologs). This would be limited to **policydir** if defined. Some system directories cannot be listed for security.
- **Policy (Script) Get Lines:** Gets a script-based policy file as an ordered array of lines, making line based modifications to the policy file easier.
- **Policy (Script) Get Full File:** Gets the full script-based policy file as a long string.
- **Policies (CSV) Get All:** Retrieves an array of CSV policies. Elements are generally strings or arrays of strings.
- **Policy (CSV) Get (by name):** Retrieves a given named CSV policy.
- **Policy (CSV) Put (by name):** Puts a given CSV policy, named on the URL.
- **Policy (Script) Set New Policy File:** Creates a new (optionally empty) policy script file. Directory is limited by **policydir** if it is set. Parent directories are not created.
- **Policy check:** Checks policy in a similar manner to **pbcheck**.
- **Get Policy file as attachment:** Retrieves a full policy file as a binary attachment.

IO Logs

- **IO Log Get:** Retrieves an IO log file. Output can be limited by **len** and **start** parameters so that individual parts of the log can be retrieved in *chunked* form.
- **IO Log List Dir:** Lists all of the files in a given directory (without checking if they are IO logs). **Filter** can be specified as a regular expression to filter output. Some system directories cannot be listed for security.
- **IO Log Get Variables:** Retrieves the log variables from the specified IO log.
- **Get IO Log file as attachment:** Retrieves an IO log file as a binary attachment.

- **IO Log search:** Searches a list of IO logs, specified with a glob style wildcard parameter **file**, for the query string **<query>**. This is a similar format to the SOLR search string in which you have a regular expression query, with **keyword:value** values. For example, **stdout:*inittab** searches for any IO logs that incorporate the word **inittab** in the output. All of the standard keyword values that can be extracted from IO logs can be used in the search criteria. Regular expression matches are not made across newlines.
- **IO Log Replay:** Retrieves and interprets an IO log file ready to be output by a GUI. Override terminal emulation using the parameter **term**, and the output can be limited by **len** and **start** parameters so that individual parts of the log can be retrieved in *chunked* form.
- **IO Log Get:** Retrieves an IO log file. Output can be limited by **len** and **start** parameters so that individual parts of the log can be retrieved in *chunked* form.
- **IO Log List Dir:** List all of the files in a given directory (without checking if they are IO logs). **Filter** can be specified as a regular expression to filter output. Some system directories cannot be listed for security.
- **IO Log Get Variables:** Retrieves the log variables from the specified IO log.
- **Get IO Log file as attachment:** Retrieves an IO log file as a binary attachment.
- **IO Log search:** Search a list of IO logs, specified with a glob style wildcard parameter **file**, for the query string **<query>**. This is a similar format to the SOLR search string where you have a regular expression query, with **keyword:value** values. For example, **stdout:*inittab** searches for any IO logs that incorporate the word **inittab** in the output. All of the standard keyword values that can be extracted from IO logs can be used in the search criteria. Regular expression matches are not made across newlines.
- **IO Log Replay:** Retrieves and interprets an IO log file ready to be output by a GUI. Override terminal emulation using the parameter **term**, and the output can be limited by **len** and **start** parameters so that individual parts of the log can be retrieved in *chunked* form.

Event Logs

- **Event Log Get:** Retrieves the specified event log.
- **Get Event Log file as attachment:** Retrieves the specified event log as a binary attachment.

Key File

- **Key Get:** Gets the specified **pb.key** file as a base64 encoded string.
- **Key Set:** Sets the specified **pb.key** to the base64 encoded string.
- **Create a new key:** Creates a new specified **pb.key** file and generates random contents.
- **Get Key file as attachment:** Retrieves the specified **pb.key** file as a binary attachment.

SOLR

- **Solr Get:** Retrieves SOLR search results based upon the supplied criteria.

Authentication

REST is a stateless protocol, which is strongly stressed and adhered to, and so each and every call must authenticate itself to the REST service. Privilege Management for Unix and Linux REST API will use an authentication method developed by Amazon Web Services that uses a pre-shared key and HMAC signature.

An Application ID, devised by the administrator, is input into a key store maintenance program that stores the ID alongside a randomly generated Application Key, and specified ACLs that specify what access the Application ID has. The Application Key is output, and these two together form the authentication mechanism.

The Application ID and a timestamp (epoch seconds) of information are hashed together using the Application Key to make an MD5 HMAC signature that is appended onto the URL of each call. It checks the timestamp to make sure it is relatively recent, retrieves the Application ID, and produces the HMAC using the same parameters to make sure the authentication is valid. The key store is encrypted by default, and can be relocated using configuration in **pb.settings**.

Installation and Configuration

The REST API is bundled in tar files and package installers.

Prerequisites

- A suitable web server that supports Fast CGI protocol, configured to support Fast CGI modules and HTTPS/SSL (with a suitable certificate). This includes the enabling of any firewalls to allow HTTPS access.
- Privilege Management for Unix and Linux v7.1.0 or above preinstalled and configured.
- Root access on the host that will provide REST API to enable the installation of the module.
- For development with the Java example sources, a Java 7 JDK is required, and the Eclipse IDE project files are provided for convenience.

The REST API must be installed on different Privilege Management for Unix and Linux hosts, based on the functionality required:

REST API Call	Privilege Management Component Requiring REST API Installation
Get License	Policy Servers
Get/Put Policy	Policy Servers
Get/Put Settings	The local host where the settings need to be read/written to
Get/Search/List iologs	Log Servers
Get Eventlogs	Log Servers
Get/Create keyfile	Policy Servers
Solr search	Any host

REST API Installed Files



Note: *tempfilepath* defines a temporary path to be used as the temporary filesystem for PMUL binaries. The default is set as */tmp*. At install time, if *pbinstall* is invoked, using *-t <tempdir>* option, *tempfilepath* is set to *<tempdir>*. *lockfilepath* defines a lock file path for PMUL binaries as needed. The default is */opt/pbul/locks*.

When REST API is installed, the following files are copied to the host:

- The binary **pbrestcall** is copied to the Administration Program Location (*/usr/sbin*, by default).
- The binary **pbconfigd** is copied to WWW server CGI directory specified during the installation.

- The sample code files are copied to the REST API sample code directory specified during the installation:
 - **examples/lighttpd/**: Example configuration files for **lighttpd**. The contents of these are referenced below in the example configuration of web server
 - **examples/java/**: Java sources and Eclipse project for Java API examples
 - **examples/java/PBULAPI/build.xml**: Ant build script to build the Java **.jar**
 - **examples/java/PBULAPI/doc**: Javadoc documentation
 - **examples/jsoncalls.html**: Example of static HTML file, containing JavaScript implementation of JSON API. The file contains JavaScript sections that implement the MD5 HMAC required to make the REST call.
 - **examples/java/PBULAPI/test**: JUnit test suite to test all areas of the JSON REST API using Java
 - **examples/java/PBULAPI/src/***: The Java example source to call the REST API
 - **examples/scripts/**: Example bash scripts that implement JSON API calls



For more information on **pbrestcall** and **pbconfigd**, please see "**pbrestcall**" on page 518 and "**pbconfigd**" on page 517.

pblighttpd Service

pblighttpd is the HTTP server daemon which launches the REST service. It is configured to run as a service in the background on hosts where a Privilege Management for Unix and Linux server component is installed. After a successful installation, the **pblighttpd** service is configured to startup and shutdown **systemd/xinetd/init** on Linux, **init** on HP-UX, **initab** for AIX, and **SMF** for Solaris 10+.

Stop pblighttpd

Platform	Command
Linux	service pblighttpd stop
HP-UX	/sbin/init.d/pblighttpd stop
AIX	stopsrc -s pblighttpd
Solaris	svcadm disable pblighttpd

When **pblighttpd** service receives a stop request, its main process will ensure that all its child processes have terminated before it exits. This feature, combined with a lock file mechanism, helps prevent a new set of processes from starting before the old set has cleaned up and exited.

Start pblighttpd

Platform	Command
Linux	service pblighttpd start
HP-UX	/sbin/init.d/pblighttpd start
AIX	startsrc -s pblighttpd
Solaris	svcadm enable pblighttpd

The **pblighttpd** service is comprised of a set of processes which is monitored by a parent process. A lock file mechanism is used to ensure that only one set exists at a time.

To avoid running out of resources during heavy load while support authentication events to a program (**eventdestinations** **authvt=|program**), **pblighttpd-svc** must be started with unlimited **NOFILES** and **NPROC**, at least. For example, on Linux with systemd:

```
# cat /etc/systemd/system/pblighttpd.service
[Unit]
Description=BeyondTrust Privilege Management - REST Server
After=network.target
[Service]
Type=forking
Environment=LD_LIBRARY_PATH=/opt/pbul/rest/lib:/opt/pbul/lib:/usr/lib/oracle/19.5/client64/lib
ExecStart=@/opt/pbul/rest/sbin/pblighttpd-svc pblighttpd-svc -d -i
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
ExecReload=/bin/kill -HUP $MAINPID
TasksAccounting=false
LimitNPROC=infinity
LimitNOFILE=infinity
LimitCORE=infinity
[Install]
WantedBy=multi-user.target
```

For legacy sysV, simply add **ulimit -u unlimited** and **ulimit -n unlimited** to the start routine of **/etc/rc.d/init.d/pblighttpd**.

Restart pblighttpd when pb.settings is changed

The **pblighttpd** service must be restarted when the following keywords are changed in **pb.settings**:

- **svccacherefresh**
- **dbsyncrefresh**
- **pblicenserefresh**
- **autofwdtime**
- **iologactioninterval**
- **sharedlibssldependencies**
- **sharedlibcurldependencies**
- **schedulingservice**
- **restservice**
- **pbrestlog**
- **messengeroutersocketpath**
- **messengerrouterqueuesize**

pblighttpd_svc.sh

pblighttpd_svc.sh is a wrapper script that can start, stop, and restart **pblighttpd** on any of the supported platforms. It is installed in the same location as the other PMUL administration programs.

Syntax

```
<prefix>pblighttpd_svc.sh<suffix> <action>
```

where *<action>* is one of the following:

- **stop:** Stop the pblighttpd service
- **start:** Start the pblighttpd service
- **restart:** Stop and then start the pblighttpd service

When used to stop the pblighttpd service, the script ensures that all the processes associated with pblighttpd have terminated before exiting and returning to the command prompt.



Note: On systems where processes take a long time to exit, the script may appear hung when it is just waiting for a child process. Use **ps** or a similar tool to list running processes and check for **pblighttpd**.

When being requested to start the pblighttpd service, the script errors out if it detects that a pblighttpd service is already running.

The script uses the pbdbutil (pbadmin) option to check if **pblighttpd** and **pbconfigd** processes are accepting requests before attempting to stop or restart.

```
<prefix>pbdbutil<suffix> --info --restsvr
```

If not accepting requests, the script displays “*The process does not seem to accept requests, attempting to <stop/restart> anyway*” to inform you of current status but proceeds with stopping or restarting the service anyway.

Privilege Management for Unix and Linux REST API Programs

pbconfigd

- **Version 8.5.0 and earlier:** **pbconfigd** setting not available
- **Version 9.0.0 and later:** **pbconfigd** setting available

The **pbconfigd** binary is utilized by the **pblighttpd** process to provide REST services. Much of the functionality that is provided by newer versions of Privilege Management for Unix and Linux uses facilities that the REST services provide. These include secure communication between services across the enterprise, a reliable replicated database layer, scheduled tasks that provide automated maintenance and housekeeping for background processes, the REST API layer used by the BeyondInsight for Unix & Linux GUI management tool, and can be used by customers to integrate Privilege Management for Unix and Linux with their own in-house processes and systems.

pbconfigd is launched by the **pblighttpd** server and communicates in JSON over the FastCGI protocol. The service is started automatically by the operating system and services requests when various other facilities of Privilege Management for Unix and Linux require it. Primary servers tend to instantiate more **pbconfigd** processes so that they can more readily serve secondary servers and clients with services such as Registration Name Service, Role Based Policy configuration, File Integrity Monitoring, and Licensing framework. Client hosts do not generally run **pbconfigd** processes, but can momentarily launch the service to allow client configuration or to retrieve valuable data for the BeyondInsight for Unix & Linux management GUI.

As more Privilege Management for Unix and Linux facilities become more reliant upon the REST service for reliable and successful functioning, it is vital to make sure that the service is working correctly. We recommend the REST log file be closely monitored and the troubleshooting guide used whenever issues are noted.

Generally **pbconfigd** is launched automatically by the **pblighttpd** service and should not be launched manually.



Note: `pbconfigd` has a `--call` option. This action requires a JSON string parameter to process and processes as if the call was made over REST. This allows specific calls that are required to action licensing and message router queuing calls to be made.

Usage

```
pbconfigd [options]
-v, --version
--help
```

Arguments

<code>-v, --version</code>	Optional. Display version and exit.
<code>--help</code>	Optional. Display this help message and exit.



For more information, please see "[pblighttpd Service](#)" on page 515.

pbrestcall

- **Version 8.0 and earlier:** `pbrestcall` setting not available
- **Version 8.1.0 and later:** `pbrestcall` setting available

The `pbrestcall` utility provides a method of making REST API calls from the command line. It calculates the MD5 HMAC of the header to provide authentication, and encodes command line parameters on the URL.



Example:

```
pbrestcall -l -a appid -k f339c33b-64d7-44f4-b113-34e344cff670
https://myhost:24351/REST/settings
```

This retrieves and display all of the `/etc/pb.settings` from the Policy Server host using the REST API, and the `-l` parameter lists them in a formatted JSON format.

Usage

```
pbrestcall [options] [param=value param=value...]
-a <appId> [<acl> ...]
-k <appKey>
-d
-X
-l
-v
```

```
-t
-p
```

The URL with HMAC and timestamp can be printed using the `-p` parameter. MD5 HMAC along with timestamp (see the setting `pbresttimeskew`) authenticates the session.

```
pbrestcall -p -l -a appid -k f339c33b-64d7-44f4-b113-34e344cff670
https://myhost:24351/REST/settings
```

```
Response
https://myhost:24351/REST/settings?appid=appid&timestamp=1620676163&hmac=9623fe94d48e50d2d600efd5
18613392
```

Arguments

-a <appId>	Mandatory. Specify Application ID to use.
-k <appKey>	Mandatory. Specify Application Key to use.
-d	HTTP body data as a string, or - to use stdin
-X	HTTP request method. GET, PUT, POST, DELETE.
-l	Long (pretty print) JSON
-v	Verbose
-t	Debug
-p	Print the URL and exit

Configuration of pbconfig FastCGI Module

The `pbconfigd` specific parameters below may need to be added to `/etc/pb.settings` manually.

restkeyencryption

- **Version 8.0 and earlier:** `restkeyencryption` setting not available
- **Version 8.1.0 and later:** `restkeyencryption` setting available

This is similar to other encryption keywords but only takes one value at a time.

```
restkeyencryption <algorithm-1>:<keyfile=/fullpath/data-file-1>
[:<startdate=yyyy/mm/dd>:<enddate=yyyy/mm/dd>]
```

**Example:**

```
restkeyencryption aes-256:keyfile=/etc/pb.key
```

Default

```
restkeyencryption aes-256
```

Used on

- All hosts where **pbconfigd** is installed



For more information, please see "[networkencryption](#)" on page 200.

pbrestlog

- **Version 8.0 and earlier:** **pbrestlog** setting not available
- **Version 8.1.0 and later:** **pbrestlog** setting available

This option configures where the **pbconfigd** binary should log debug and error messages.

**Example:**

```
pbrestlog /var/log/pbrest.log
```

Default

Depending on the operating system standards, this can be any of the following:

- **/var/log/pbrest.log**
- **/var/adm/pbrest.log**
- **/usr/adm/pbrest.log**

Used on

All hosts where **pbconfigd** is installed

pbrestkeyfile

- **Version 8.0 and earlier:** **pbrestkeyfile** setting not available
- **Version 8.1.0 and later:** **pbrestkeyfile** setting available

This option allows the specification of where the **pbconfigd** keys are kept. This keystore is encrypted using the **restkeyencryption** option above. It may be written as an absolute path, or a path relative to the **basedir** setting.



Example:

```
pbrestkeyfile /mypath/pbrstkeys.db
```

Default

```
pbrestkeyfile /opt/pbul/dbs/pbrstkeys.db
```

Used on

All hosts where **pbconfigd** is installed

REST ACL's

To allow different applications to have different access over the REST API, ACLs have been implemented and can be created against given application ID's. These take the form of regular expressions that match specific REST API URLs. Each application ID can have up to 8 regular expressions assigned to it.

Examples

GET:/settings	Allow retrieval of any settings
PUT:/settings	Allow the change of any settings
\(GET PUT\)\/setting\/(submitmaster acceptmasters\)	Allow the modification of submitmaster and acceptmasters
GET:/iolog/.file=\%2Ftmp\%2F.*	Allow the retrieval of any iolog in /tmp

Configuration of pbconfigd Keystore

To enable the authentication of REST API sessions pre-shared keys are created with application identifiers. These are used to identify the application and to create a MD5 HMAC (along with a timestamp) to authenticate the session.



Example: The following is an example of the most basic configuration:

```
pbdbutil --rest -g appid
```

This creates an application ID of **appid** and displays the application key, which needs to be noted, as it cannot be retrieved and added to the web application.

Specify a REST URL Path and Parameters

RESTful interfaces use the URL path to specify data and attributes, not actions. Actions are specified using the HTTP request methods **GET**, **PUT**, **POST**, and **DELETE**. Data is grouped together in the URL path, so for example all Privilege Management for Unix and Linux settings are specified in the path **/REST/settings** (or **/REST/setting** when accessing a single setting). HTTP **GET** requests always pass parameters (URL encoded) as URL parameters on the URI. **PUT**, **POST**, and **DELETE** can use either the URI or the HTTP body to pass more complicated data.



Example: To retrieve the **iolog/tmp/iolog+root.log**:

```
GET https://<host>:<port>/REST/iolog?file=%2f/tmp%2fiolog%2broot%2elog
```



Example: To overwrite the **pb.settings** parameter **submitmaster**:

```
PUT https://<host>:<port>/REST/setting/submitmaster  
{ "setting": {"values":["pbuild", "pbuild2", "pbuild3"]}}
```

REST Calls and Parameters

Authentication

Authentication (unsuccessful)

Invalid authentications can occur because of incorrect **appid**, **appkey**, invalid HMAC calculation, or invalid **pbrest** keystore.

```
GET https://pbuild:24351/REST/setting/policydir?appid=<appid>&timestamp=<timestamp>&hmac=<hmac>
```

```
RESPONSE {"error":"8119 Invalid Authentication for appId 'myappid' from  
192.168.16.128","status":8119}
```

Settings

Get Settings

Get all of the settings from **pb.settings** (or equivalent). Each setting has one of 4 distinct types:

- String
- Boolean
- List of Strings
- **altsubmitmasters**: Has a special list of Privilege Management for Unix and Linux objects.

```
GET https://pbuild:24351/REST/settings?appid=<appid>&timestamp=<timestamp>&hmac=<hmac>
```

```
RESPONSE {"status":0,"settings":[{"values":["8da3e912","83099374","52adfcae","27825f05"],  
"description":"Validation","name":"validation","type":3},  
{"description":"Licensing Data to Save","name":"licensedata","value":"datenodename","type":2},  
...}
```

Get Setting

Gets an individual setting as specified on the URL.

```
GET https://pbuild:24351/REST/setting/submitmasters?appid=<appid>&timestamp=  
<timestamp>&hmac=<hmac>
```

```
RESPONSE {"setting":[{"values":["pbuild","pbuild2","pbuild3"],"description":"Submit  
Masters","name":"submitmasters","type":3}],"status":0}
```

Put Setting

Put (modify) a setting into the **pb.settings** file. The type needs to correspond to the original setting type.

```
PUT https://pbuild:24351/REST/setting/warnuseronerror?appid=<appid>&timestamp  
=<timestamp>&hmac=<hmac>
```

```
REQUEST {"setting":{"name":"warnuseronerror","value":"no","type":1}}
```

```
RESPONSE {"status":0}
```

Get Settings file as attachment

Retrieves the whole **pb.settings** file as a binary attachment.

```
GET https://pbuild:24351/REST/settingsfile?appid=<appid>&timestamp=  
<timestamp>&hmac=<hmac>&file=%2Fetc%2Fpb.settings <binary attachment>
```

License

List Client License Information

List Client Host Details

Optional argument: `verbose=0|1`

```
GET https://pbuild:24351/pbrest/REST/v2.0/license/clients?appid=<appid>&timestamp=
<timestamp>&hmac=<hmac>
```

Without verbose:

```
RESPONSE { "clients" : [ {"lid":123, "uuid":"66fd4906-81c9-4739-8a27-308546af90ae",
"fqdn":"myhost1.com", "addr" : "[ {"family\" : 4, \"port\" : 5678, \"addr\" : \"192.168.1.1\" }]",
"lastupdated" : "2017-08-01", "retired" : "2017-08-01", "recycle" : "2017-09-01" }]}]
```

With verbose:

```
RESPONSE { "clients" : [
{"lid":123, "uuid":"66fd4906-81c9-4739-8a27-308546af90ae", "fqdn":"myhost1.com", "addr" : "[
{"family\" : 4, \"port\" : 5678, \"addr\" : \"192.168.1.1\" }]", "lastupdated" : "2017-08-01",
"retired" : "2017-08-01", "recycle" : "2017-09-01", "stats" : [ { "lid" : 123, "svc" :
"pbpolicy", "firstupdated" : "2017-01-01", "lastupdated" : "2017-10-01"},{ "lid" : 123, "svc" :
"sudopolicy", "firstupdated" : "2017-01-01", "lastupdated" : "2017-11-01"},{ "lid" : 123, "svc" :
"fim", "firstupdated" : "2017-01-01", "lastupdated" : "2017-11-21"}]}]
```

List Client Host Details

Optional argument: `verbose=0|1`

```
GET https://pbuild:24351/pbrest/REST/v2.0/license/clients/<host>?appid=<appid>
&timestamp=<timestamp>&hmac=<hmac>
```

RESPONSE

As above, but only for one specified host.

List Client Host (wildcard) Details

Optional argument: `verbose=0|1`

```
GET https://pbuild:24351/pbrest/REST/v2.0/license/clients?appid=<appid>&timestamp=
<timestamp>&hmac=<hmac>&hostname=<wildcard>
```

RESPONSE

As above, but only for wildcarded hosts.

List Service License Information

List All Service Records

Optional argument: `verbose=0|1`

```
GET https://pbuild:24351/pbrest/REST/v2.0/license/svc?appid=<appid>&timestamp=
<timestamp>&hmac=<hmac>
```

Without verbose:

```
RESPONSE {"stats" : [ {"attr" : "PBULPolClnts", "cnt" : 800}, {"attr" : "SudoPolClnts", "cnt" :
800}, {"attr" : "ACAClnts", "cnt" : 800}]}
```

With verbose:

```
RESPONSE {"stats" : [ { "lid" : 123, "attr" : "PBULPolClnts", "firstupdated" : "2017-01-01",
"lastupdated" : "2017-10-01"}, { "lid" : 123, "attr" : "SudoPolClnts", "firstupdated" : "2017-01-
01", "lastupdated" : "2017-11-01"}, { "lid" : 123, "attr" : "ACAClnts", "firstupdated" : "2017-01-
01", "lastupdated" : "2017-11-21"} ]}}
```

List Specified Service Records

Optional argument: `verbose=0|1`

```
GET https://pbuild:24351/pbrest/REST/v2.0/license/svc/<svctype>?appid=<appid>&timestamp=
<timestamp>&hmac=<hmac>
```

Without verbose:

```
RESPONSE {"stats" : [ {"attr" : "PBULPolClnts", "cnt" : 800}]}
```

With verbose:

```
RESPONSE {"stats" : [ { "lid" : 123, "attr" : "PBULPolClnts", "firstupdated" : "2017-01-01",
"lastupdated" : "2017-10-01"} ]}}
```

Get License Record

Optional argument: `verbose=0|1`

```
GET https://pbuild:443/pbrest/REST/v2.0/license?appid=<appid>&timestamp=<timestamp>&hmac=<hmac>
```

```
RESPONSE { "license" : { "PBULPolClnts":100, "SudoPolClnts":0, "RBPClnts":100, "ACAClnts":100, "AKAClnts":0, "FIMClnts":10, "SOLRClnts":2, "Expires":"2018/12/25 23:59:00", "Terminates":"2019/03/25 23:59:00", "HostId":"6fce4d59-7359-4c7e-a793-90f3989214a0", "Owner":"My Company PLC", "Comment":"This is the PBUL license for My Company", "AutoExpiry": 90, "Recycle": 30, "HMAC":"skSUgmkZ491x6Bzul2g5wgl/WsY6WbFUwMB8kT3zmro="}}
```

Put License Record

```
PUT https://pbuild:443/pbrest/REST/v2.0/license?appid=<appid>&timestamp=<timestamp>&hmac=<hmac>
```

Args

```
{ "whoami" : "root", "license" : { "PBULPolClnts":100, "SudoPolClnts":0, "RBPClnts":100, "ACAClnts":100, "AKAClnts":0, "FIMClnts":10, "SOLRClnts":2, "Expires":"2018/12/25 23:59:00", "Terminates":"2019/03/25 23:59:00", "HostId":"6fce4d59-7359-4c7e-a793-90f3989214a0", "Owner":"My Company PLC", "Comment":"This is the PBUL license for My Company", "AutoExpiry": 90, "Recycle": 30, "HMAC":"skSUgmkZ491x6Bzul2g5wgl/WsY6WbFUwMB8kT3zmro="}}
```

Failure

```
RESPONSE {"status": 3828,"error": "3828.41 Failed to update License - Invalid argument"}
```

Retire Client Record

```
PUT https://pbuild:443/pbrest/REST/v2.0/license/retire?appid=<appid>&timestamp=<timestamp>&hmac=<hmac>
```

By UUID

```
{ "whoami" : "root", "retire" : { "uuid" : "..." }}
```

By FQDN

```
{ "whoami" : "root", "retire" : { "fqdn" : "..." }}
```

By List

```
{ "whoami" : "root", "retire" : [ {"uuid" : "..." }, {"uuid" : "..." } ] }
```



```
RESPONSE {"status" : 0}
```

Process licensing write queues and send them to the primary license server:

```
PUT put_sync cd /usr/lib/beyondtrust/pb/rest/sbin
./pbconfigd --call '{ "appid" : "appid", "appkey" : "866bfc75-72d2-4dcf-9b08-4dbd9474a493",
"request" : {"content_type" : "application/json", "method" : "GET", "uri" : "/v2/license/put_
sync" } }'
```

```
RESPONSE Status: 200
Content-type: application/json
{"status":0}
```

Retrieve licensing statistics from the primary license server and write them to the local pblicense.db:

```
GET get_sync cd /usr/lib/beyondtrust/pb/rest/sbin
./pbconfigd --call '{ "appid" : "appid", "appkey" : "866bfc75-72d2-4dcf-9b08-4dbd9474a493",
"request" : {"content_type" : "application/json", "method" : "GET", "uri" : "/v2/license/get_
sync" } }'
```

```
RESPONSE Status: 200
Content-type: application/json
{"status":0}
```

Policy

Policy List Dir

List all of the files in a given directory (without checking they are policy files). Some system directories cannot be listed for security.

```
GET https://pbuild:24351/REST/policies?appid=<appid>&timestamp=<timestamp>&hmac=<hmac>
&path=%2Fopt%2Fpbul%2Fpolicies
```

```
RESPONSE {
  "dir": [
    {
      "path": "/opt/pbul/policies/pbul_policy.conf",
      "type": "file",
      "name": "pbul_policy.conf",
      "size": 5345,
      "mtime": "2018-11-02 16:36:23",
      "where": "fs"
    },
    {
      "path": "/opt/pbul/policies/pb.conf",
      "type": "file",
      "name": "pb.conf",
      "size": 228,
      "mtime": "2018-11-17 16:20:55",
      "where": "fs"
    },
    {
      "path": "/opt/pbul/policies/pbul_functions.conf",
      "type": "file",
      "name": "pbul_functions.conf",
      "size": 11747,
      "mtime": "2018-11-02 16:36:23",
      "where": "fs"
    }
  ]
}
```

Policy (Script) Get Lines

Get a script based policy file as ordered an array of lines, making line based modifications to the policy file easier.

```
GET https://pbuild:24351/REST/policies?appid=<appid>&timestamp=<timestamp>
&hmac=<hmac>&format=script&file=%2Fopt%2Fpbul%2Fpolicies%2Fpb.conf
```

```
RESPONSE {
  "file": "/opt/pbul/policies/pb.conf",
  "format": "script",
  "lines": [
```

```

"result=getuserpasswd(user, \"Passwd for \"+user+": \", 1, \"/opt/pbul/gp001\", 20);",
"printf(\"result=%d\\n\\n\", result);",
"if (result == 0) ",
"reject;",
"else",
"accept;"
]
}

```

Policy (Script) Get Full File

Get the full script based policy file as a long string.

```

GET https://pbuild:24351/REST/policy?appid=<appid>&timestamp=<timestamp>
&hmac=<hmac>&format=script&file=%2Fopt%2Fpbul%2Fpolicies%2Fpb.conf

```

```

RESPONSE {
  "file": "/opt/pbul/policies/pb.conf",
  "format": "script",
  "policy": "result=getuserpasswd(user, \"Passwd for \"+user+": \", 1, \"/opt/pbul/gp001\",
20);\nprintf(\"result=%d\\n\\n\", result);\nif (result == 0) \n reject;\nelse\n accept;\n"
}

```

Policies (CSV) Get All

Retrieves an array of CSV policies. Elements are generally strings or arrays of strings.

```

GET https://pbuild:24351/REST/policies?appid=<appid>&timestamp=<timestamp>
&hmac=<hmac>&format=csv&file=%2Fetc%2Fpb%2Fpb.csv

```

```

RESPONSE {"status":0,"file":"/etc/pb/pb.csv","format":"csv","policies":
[{"dateend":"none","enabled":"Active","verifyuser":0,"adgrps":
["PBSE\\pbqa","PBSE\\pbdev"],"datestart":"none","timeoutstop":"","hostsmatch":"1","args":
["0","0","0","0","0","0"],"lclgrps":["root","pbdev"],"subhosts":["ANY"],"adusers":
[""],"type":"Accept","runcmds":
["","","","","",""],"hostlistsmatch":"1","runhosts":[""],"subcmds":

["bash","csh","ksh","ksh93","tcsh","sh"],"defineenv":0,"name":"Shell","timestart":"none","timeend":
"none","keylog
":0,"preserveenv":0,"runas":["root","qareveal","PBSE\\qareveal","qareveal@pbse.lab"],"lcllusers":
["ctaylor"]},
{"dateend":"none","enabled":"Active","verifyuser":0,"adgrps":
["PBSE\\pbqa","PBSE\\pbdev"],"datestart":"none","timeoutstop":"","hostsmatch":"1","args":
["0","0","0","0","0","0"],"lclgrps":["root","pbdev"],"subhosts":["ANY"],"adusers":
[""],"type":"Accept","runcmds":
["","","","","",""],"hostlistsmatch":"1","runhosts":[""],"subcmds":

["bash","csh","ksh","ksh93","tcsh","sh"],"defineenv":0,"name":"FOO","timestart":"none","timeend":
"none","keylog

```

```
":0,"preserveenv":0,"runas":["root","qareveal","PBSE\\qareveal","qareveal@pbse.lab"],"lcllusers":
["ctaylor"]}, ...
```

Policy (CSV) Get (by name)

Retrieve a given named CSV policy.

```
GET https://pbuild:24351/REST/policy/BOO?appid=<appid>&timestamp=<timestamp>
&hmac=<hmac>&format=csv&file=%2Fetc%2Fpb%2Fpb.csv
```

```
RESPONSE {"status":0,"file":"/etc/pb/pb.csv","policy":
{"dateend":"none","enabled":"Active","verifyuser":0,"adgrps":
["PBSE\\pbqa","PBSE\\pbdev"],"datestart":"none","timeoutstop":"","hostsmatch":"1","args":
["0","0","0","0","0","0"],"lclgrps":["root","pbdev"],"subhosts":["ANY"],"adusers":
[""],"type":"Accept","runcmds":
["","","","","",""],"hostlistsmatch":"1","runhosts":[""],"subcmds":

["bash","csh","ksh","ksh93","tcsh","sh"],"defineenv":0,"name":"BOO","timestart":"none","timeend":
"none","keylog
":0,"preserveenv":0,"runas":["root","qareveal","PBSE\\qareveal","qareveal@pbse.lab"],"lcllusers":
["ctaylor"]},"format":"csv"}
```

Policy (CSV) Put (by name)

Put a given CSV policy, named on the URL.

```
PUT https://pbuild:24351/REST/policy/BOO?appid=<appid>&timestamp=<timestamp>
&hmac=<hmac>&format=csv&file=%2Fetc%2Fpb%2Fpb-tmp.csv
```

```
REQUEST {"policy":{"dateend":"none","enabled":"disabled","verifyuser":0,"adgrps":
["PBSE\\pbqa","PBSE\\pbdev"],"datestart":"none","timeoutstop":"","hostsmatch":"1","args":
["0","0","0","0","0","0"],"lclgrps":["root","pbdev"],"subhosts":["ANY"],"adusers":
[""],"type":"Accept","runcmds":
["","","","","",""],"hostlistsmatch":"1","subcmds":
["bash","csh","ksh","ksh93","tcsh","sh"],"runhosts":

[""],"defineenv":0,"name":"BOO","keylog":0,"timeend":"none","timestart":"none","preserveenv":0,"r
unas":
["root","qareveal","PBSE\\qareveal","qareveal@pbse.lab"],"lcllusers":["ctaylor"]}}
```

```
RESPONSE {"status":0}
```

Policy (Script) Set New Policy File

Create a new (optionally empty) policy script file. Directory is limited by **policydir** if it is set.

```
POST https://pbuild:24351/REST/policy?appid=<appid>&timestamp=<timestamp>
&hmac=<hmac>&format=script&file=%2Fetc%2Fpb%2Ffoobar
```

```
REQUEST {"script":"accept;\n"}
```

```
RESPONSE {"status":0,"file":"/etc/pb/foobar"}
```

Policy check (unsuccessful)

Checks policy in a similar manner to **pbcheck**.

```
GET https://pbuild:24351/REST/policies/check?appid=<appid>&timestamp=<timestamp>
&hmac=<hmac>&file=%2Fopt%2Fpbul%2Fpolicies%2FpbOLD.conf
```

```
RESPONSE {
  "status": 8103,
  "error": "8103.1 Error parsing policy file /opt/pbul/policies/pbOLD.conf, 3964 file
/opt/pbul/policies/pbOLD.conf does not exist"
}
```

Policy check (successful)

Checks policy in a similar manner to **pbcheck**.

```
GET https://pbuild:24351/REST/policies/check?appid=<appid>&timestamp=<timestamp>
&hmac=<hmac>&file=%2Fetc%2Fpb%2Ftry
```

```
RESPONSE {"message":"Syntax check completed with no problems detected","status":0}
```

Policy check inline script (unsuccessful)

Checks inline script policy in a similar manner to **pbcheck**.

```
PUT https://localhost:24351/REST/policy/check?appid=<appid>&timestamp=<timestamp>&hmac=<hmac>
```

```
REQUEST { "script" : "foobar\nbarfoo\n" }
```

```
RESPONSE {"errors":[{"line":1,"file":"inline","msg":"syntax error, unexpected $end"},
{"line":1,"file":"inline","msg":"1167.2 Expected a statement"}],"status":8103,"error":"8103.1
Error parsing policy script"}
```

Policy check inline script (successful)

Checks inline script policy in a similar manner to **pbcheck**.

```
PUT https://localhost:24351/REST/policy/check?appid=<appid>&timestamp=<timestamp>&hmac=<hmac>
```

```
REQUEST { "script" : "accept;" }
```

```
RESPONSE{"message":"Syntax check completed with no problems detected"}
```

Get Policy file as attachment

Retrieves a full policy file as a binary attachment.

```
GEThttps://pbuild:24351/REST/policyfile?appid=<appid>&timestamp=<timestamp>&hmac=<hmac>&file=%2Fetc%2Fpb%2Fpb.conf <binary attachment>
```

Role Based Policy Authentication

Test Role Based Policy authentication.

```
PUT https://localhost:24351/REST/policy/rbp/checkauth?appid=<appid>&timestamp=<timestamp>&hmac=<hmac>
```

Parameters

```
{ "rbp" : { "user" : "root", "submithost" : "pbuild", "command" : "/usr/bin/id", "runhost": "pbuild1", "pbclientmode": "pbrun" }}
```

The parameter node must contain at least **user**, **submithost**, and **command**, but may also contain any other Privilege Management for Unix and Linux policy variable, used when matching roles. On a positive response, the **info** part of the JSON response is the role **row** that matched.



Example: Positive Response

```
{
  "result": {
    "access": "allowed",
    "iolog": "/tmp/admin_iolog_root_XXXXXX",
    "userMessage": "hello root - risk 9\n",
    "info": {
      "name": "Admin",
      "runuser": "root",
    }
  }
}
```



```
"runhost": "pbuild1",
"risk": 9,
"action": "A",
"iolog": "/tmp/admin_iolog_%user%_XXXXXX",
"message": "hello %user% - risk %pbrisklevel%",
"variables": null,
"auth": null,
"script": null,
"runcommand": ""
}
}
```



Example: Negative Response

```
{
  "result": {
    "access": "denied"
  }
}
```

IO Logs

I/O Log Get

Retrieves an I/O log file. Output can be limited by **len** and **start** parameters so that individual parts of the log can be retrieved in *chunked* form.

```
GET https://pbuild:24351/REST/iolog?appid=<appid>&timestamp=<timestamp>
&hmac=<hmac>&file=%2Ftmp%2Fiolog.root.ckYxun
```

```
RESPONSE {"bytes":5832,"start":-1,"status":0,"file":"/tmp/iolog.root.ckYxun","len":0,"log":
[{"time":1386255179},
{"cols":80,"cmd":"CMD_WINCH","rows":24}, {"time":1386255179},
{"blob":"G10wO2N0YXlsb3JAcGJ1aWxkOi9ob211L2N0YXlsb3IHG1s/MTAzNGhbcm9vdEBwYnVpbGQgY3RheWxvcl0jIA==", "cmd":"stdout","blen":64}, {"time":1386255180}, {"blob":"bA==", "cmd":"stdin","blen":1},
{"time":1386255180}, {"blob":"bA==", "cmd":"stdout","blen":1}, {"time":1386255180},
{"blob":"cw==", "cmd":"stdin","blen":1}, {"time":1386255180},
{"blob":"cw==", "cmd":"stdout","blen":1},
{"time":1386255181}, {"blob":"DQ==", "cmd":"stdin","blen":1}, {"time":1386255181},
{"blob":"DQo=", "cmd":"stdout","blen":2}, {"time":1386255181}, ...
```

I/O Log List Dir

List all the files in a given directory (without checking if they are I/O logs). **Filter** can be specified as a regular expression to filter output. Some system directories cannot be listed for security.

```
GET https://pbuild:24351/REST/iolog?appid=<appid>&timestamp=<timestamp>&hmac=<hmac>&path=%2Ftmp
```

```
RESPONSE {"dir":
[{"name":"iolog.root.XXXXXX","mtime":1386252738,"path":"/tmp/iolog.root.XXXXXX","size":4928},
{"name":".8.0.0-
04.debug.60505","mtime":1386332287,"path":"/tmp/.8.0.0-04.debug.60505","size":850},
{"name":"pbcheck.log","mtime":1386076851,"path":"/tmp/pbcheck.log","size":94},
{"name":"iolog.root.jvruz1","mtime":1386255133,"path":"/tmp/iolog.root.jvruz1","size":5032},
{"name":".8.0.0-
04.debug.42263","mtime":1386335007,"path":"/tmp/.8.0.0-04.debug.42263","size":46},
{"name":".8.0.0-
04.debug.42219","mtime":1386334929,"path":"/tmp/.8.0.0-04.debug.42219","size":835},
{"name":"iolog.root.0HEKM5","mtime":1386256245,"path":"/tmp/iolog.root.0HEKM5","size":4979}, ...
```

I/O Log Get Variables

Retrieves the log variables from the specified I/O log.

```
GET https://pbuild:24351/REST/iolog/variables?appid=<appid>&timestamp=<timestamp>
&hmac=<hmac>&file=%2Ftmp%2Fiolog.root.ckYxun
```



```
RESPONSE {"status":0,"log":{"umask":18,"masterlocale":"en_US",
"nice":0,"rungroup":"root","eventlog":"/var/log/800pb.eventlog",
"loghostip":"192.168.16.138",
"masterhostip":"192.168.16.138",
"rlimit_fsize":-1,
"pbmasterdsysname":"Linux",
"optopt":"","time":"14:52:58",
"submithost":"pbuild",
"event":"Accept",
"pblogdversion":"#1 SMP Wed Oct 16 18:37:12 UTC 2013",
>false":0,
"runrlimit_core":0,
"year":2013,
"groups":
[[],
"pbversion":"8.0.0-06",
"pbmasterdversion":"#1 SMP Wed Oct 16 18:37:12 UTC 2013",
"host":"pbuild",
"optind":1,
"pbmasterdnodename":"pbuild",
"rlimit_cpu":-1,
"logservers":
[],
"runrlimit_memlock":65536,
"runbkgd":0,
"rcsworkgroup":"tramboyoPBULMasterBeyondTrustWorkgroup",
"rungroups":
[],
"logserverlocale":"en_US",
"runeffectiveuser":"root",
"pbrunmachine":"x86_64",
"runtimeout":0,
"month":12,
"dayname":"Thu",
"pbrunnodename":"pbuild",
"runoptimizedrunmode":1,
"ilog":"/tmp/iolog.root.XXXXXX",
"submitlocale":"en_GB.UTF-8",
"runrlimit_nofile":1024,
"optarg":"","cwd":"/home/ctaylor",
"runrlimit_cpu":-1,
"date":"2013/12/05", ...
```

Get I/O Log File as Attachment

```
GET https://pbuild:24351/REST/iologfile?appid=<appid>&timestamp=<timestamp>
&hmac=<hmac>&file=%2Ftmp%2Fiolog.root <binary attachment>
```

I/O Log Search

Search a list of logs, specified with a glob style wildcard parameter **file**, for the query string **<query>**. This is a similar format to the SOLR search string where you have a regular expression query, with **keyword:value** values. For example **stdout:*inittab** searches for any I/O logs that incorporate the word **inittab** in the output. All of the standard keyword values that can be extracted from I/O logs can be used in the search criteria. Regular expression matches are not made across newlines.

```
GET https://pbuild/REST/iolog/search?appid=<appid>&timestamp=<timestamp>
&hmac=<hmac>&file=%2Ftmp%2Fiolog%2a&query=stdout%3a%2e%2apbrest%2e
```

```
RESPONSE {"iologs":["/tmp/iolog.root.LNRRnt",
"/tmp/iolog.root.XXXXXX"],
"status":0}
```

I/O Log Get Replay

Retrieves and interprets an I/O log file ready to be output by a GUI. Terminal emulation can be overridden using the parameter **term**, and the output can be limited by **len** and **start** parameters so that individual parts of the log can be retrieved in *chunked* form.

```
GET https://pbuild:24351/REST/iolog/replay?appid=<appid>&timestamp=<timestamp>
&hmac=<hmac>&file=%2Ftmp%2Fiolog.root.ckYxun
```

```
RESPONSE {"status":0,"file":"/tmp/iolog-root.HxV7YJ",
"term":"xterm",
"start":-1,
"len":0,
"bytes":10676,
"log":
[{"type":"token",
"name":"sz",
"values":[44,125]},
{"type":"text",
"time":1398357936,
"value":"[root@pbuild tmp]# "},
{"type":"text",
"time":1398357937,
"value":"1"},
{"type":"text",
"time":1398357937,
"value":"s"},
{"type":"token",
"name":"absx",
"values":[0]},
{"type":"token",
"name":"nel"}]
```

```
{
  "type": "text", "time": 1398357937, "value": "beyondtrust_pbinstall"},
  {"type": "token", "name": "htab", "values": [1]},
  {"type": "token", "name": "htab", "values": [1]}, {"type": "token", "name": "htab", "values": [1]},
  {"type": "token", "name": "htab", "values": [1]}, {"type": "text", "time": 1398357937, "value": "img2c.8.5.0-01.debug.31350 img2c.8.5.0-01.debug.31548"}, {"type": "token", "name": "absx", "values": [0]},
  {"type": "token", "name": "nel"}, {"type": "text", "time": 1398357937, "value": "beyondtrust_Solrinstall.prompt_log"},
  {"type": "token", "name": "htab", "values": [1]}, {"type": "token", "name": "htab", "values": [1]},
  {"type": "text", "time": 1398357937, "value": "img2c.8.5.0-01.debug.31359 img2c.8.5.0-01.debug.31557"},
  {"type": "token", "name": "absx", "values": [0]}, {"type": "token", "name": "nel"},
  {"type": "text", "time": 1398357937, "value": "beyondtrust_Solrinstall.prompt_log.ctime.Dec_5_14:15 img2c.8.5.0-01.debug.31368 iolog-root.HxV7YJ"}, {"type": "token", "name": "absx", "values": [0]},
  {"type": "token", "name": "nel"},
  {"type": "text", "time": 1398357937, "value": "beyondtrust_Solrinstall.prompt_log.ctime.Dec_5_14:16 img2c.8.5.0-01.debug.31377 keyring-4E6Ccd"}, {"type": "token", "name": "absx", "values": [0]},
  {"type": "token", "name": "nel"},
  {"type": "text", "time": 1398357937, "value": "beyondtrust_Solrinstall.prompt_log.ctime.Dec_5_14:18 img2c.8.5.0-01.debug.31386 keyring-IAzgb5"}, {"type": "token", "name": "absx", "values": [0]},
  {"type": "token", "name": "nel"},
  {"type": "text", "time": 1398357937, "value": "beyondtrust_Solrinstall.prompt_log.ctime.Dec_5_14:21 img2c.8.5.0-01.debug.31395 keyring-qafDnO"}, {"type": "token", "name": "absx", "values": [0]},
  {"type": "token", "name": "nel"},
  {"type": "text", "time": 1398357937, "value": "gedit.ctaylor.233387052"},
  {"type": "token", "name": "htab", "values": [1]},
  {"type": "token", "name": "htab", "values": [1]}, {"type": "token", "name": "htab", "values": [1]},
  {"type": "token", "name": "htab", "values": [1]}, {"type": "text", "time": 1398357937, "value": "img2c.8.5.0-01.debug.31404 keyring-sQMK20"}, {"type": "token", "name": "absx", "values": [0]},
  {"type": "token", "name": "nel"},
  {"type": "text", "time": 1398357937, "value": "gedit.root.2441861592"},
  {"type": "token", "name": "htab", "values": [1]},
  {"type": "token", "name": "htab", "values": [1]}, {"type": "token", "name": "htab", "values": [1]}, ...
}
```

I/O Log Cached List

If you chose to install the Log Cache database for use with PowerBroker Server Management Console, the log host creates and maintains a database to cache the names, locations, and other pertinent information about I/O logs. You can retrieve the list of cached I/O log information using the following calls. Options are provided to filter and sort the output based on certain criteria. Output can also be limited by **len** and **start** parameters so that individual parts of the log can be retrieved in *chunked* form.

```
GET https://pbuild/REST/iolog/list?appid=<appid>&timestamp=<timestamp>&hmac=<hmac>
```

```
RESPONSE {"result":
  [{"uniqueid": "ac14202058221f4b116C", "partnum": 1, "createtime": 1478631243787, "loghost": "uni.symark.com", "logpath": "/var/log/pbiolog_SEPI7FJ5b", "submithost": "uni.symark.com", "submituser": "root", "runhost": "uni.symark.com", "runuser": "root", "runcmd": "rm"},
  {"uniqueid": "ac14202058221fb9118E", "partnum": 1, "createtime": 1478631354266, "loghost": "uni.symark.com", "logpath": "/var/log/pbiolog_SEPWd7Drt", "submithost": "uni.symark.com", "submituser": "root", "runhost": "uni.symark.com", "runuser": "root", "runcmd": "passwd"}]}
```

Filters

To retrieve the cached I/O log list filtered by the log path name, use the **REST GET HTTP** method with a URL similar to **https://.../list?path**, specifying a glob wildcard.



Example:

```
https://pbuild/REST/iolog/list?appid=<appid>&timestamp=<timestamp>&hmac=<hmac>&path=%2fva  
r%2flog%2f%2apb%5fiolog%2a
```

To retrieve the cached list of I/O log containing events on or after the given date/time, use the **REST GET HTTP** method with a URL similar to **https://.../list?from**, specifying date/time in the **yyyy-mm-dd HH:MM** format.



Example:

```
https://pbuild/REST/iolog/list?appid=<appid>&timestamp=<timestamp>&hmac=<hmac>&from=2016%  
2d11%2d02+18%3a00
```

To retrieve the cached list of I/O log containing events on or before the given date/time, use the **REST GET HTTP** method with a URL similar to **https://.../list?to**, specifying date/time in the **yyyy-mm-dd HH:MM** format.



Example:

```
https://pbuild/REST/iolog/list?appid=<appid>&timestamp=<timestamp>&hmac=<hmac>&to=2016%2d  
11%2d02+18%3a00
```

To retrieve the cached I/O log list filtered by **runhost**, use the **REST GET HTTP** method with a URL similar to **https://.../list?runhost**, specifying a **runhost**.



Example:

```
https://pbuild/REST/iolog/list?appid=<appid>&timestamp=<timestamp>&hmac=<hmac>&runhost=un  
i%2esymark%2ecom
```

To retrieve the cached I/O log list filtered by **submithost**, use the **REST GET HTTP** method with a URL similar to **https://.../list?submithost**, specifying a **submithost**.



Example:


```
https://pbuild/REST/iolog/list?appid=<appid>&timestamp=<timestamp>&hmac=<hmac>&submithost  
=uni%2esymark%2ecom
```

To retrieve the cached IO log list filtered by **submituser**, use the **REST GET HTTP** method with a URL similar to **https://.../list?submituser**, specifying a **submituser**.

 **Example:**


```
https://pbuild/REST/iolog/list?appid=<appid>&timestamp=<timestamp>&hmac=<hmac>&submituser=root
```

To retrieve the cached I/O log list filtered by **runuser**, use the **REST GET HTTP** method with a URL similar to **https://.../list?runuser**, specifying a **runuser**.

 **Example:**


```
https://pbuild/REST/iolog/list?appid=<appid>&timestamp=<timestamp>&hmac=<hmac>&runuser=root
```

To retrieve the cached I/O log list filtered by **runcommand**, use the **REST GET HTTP** method with a URL similar to **https://.../list?runcmd**, specifying a **runcommand**.

 **Example:**

```
https://pbuild/REST/iolog/list?appid=<appid>&timestamp=<timestamp>&hmac=<hmac>&runcmd=rm
```

To retrieve the cached I/O log list limited by an offset and number of records, use the **REST GET HTTP** method with a URL similar to **https://.../list?path**, specifying **start** and/or **len** parameters.

 **Example:**

```
https://pbuild/REST/iolog/list?appid=<appid>&timestamp=<timestamp>&hmac=<hmac>&start=10&len=5
```

Event Logs

Get Event Destinations

Retrieve a list of the current Event Log destinations. This details only the current settings and includes all the current options. A "null" or "false" attribute is taken as disabled.

```
GET
https://<host>:24351/REST/v2.0/events/destinations?appid=<appid>&timestamp=<timestamp>&hmac=<hmac>
```

```
pbrestcall -a <appid> -k <key> -l -X GET https://<host>:24351/REST/v2/events/destinations
```

```
Response
{
  "destinations":
  {
    "db": "/var/log/pb.eventlog.db",
    "ff": "/tmp/pb.eventlog",
    "prog": null,
    "odbc": "MySQL",
    "syslog": false
  }
}
```

Event Log Get

Retrieves the specified event log.

```
GET https://<host>:24351/REST/v2.0/events?appid=<appid>&timestamp=<timestamp>&hmac=<hmac>
```

```
pbrestcall -a <appid> -k <key> -l -X GET https://<host>:24351/REST/v2/events
```

```
Response
{
  "events": [
    {
      "recnum": 2,
      "uniqueid": "c0a8108a5d0cce5b7239",
      "etype": "A",
      "epoch": "2019-06-21 18:03:52",
      "submituser": "root",
      "submithost": "pbuild",
      "runuser": "root",
      "runhost": "pbuild",
      "runcommand": "id",
      "exitstatus": null
    }
  ]
}
```

Filtered/Paged Events

Retrieve events from the data source, providing paging and filter attributes. This will work slowly on flat-file event files as each event is searched from the first to the limit (or end) sequentially. However, it should be instantaneous on database and ODBC sources.

from	Match events after/from <date> (in time_t epoch or YYYY/MM/DD or full "YYYY/MM/DD HH:MM:SS" format)
to	Match events before/to <date> (in time_t epoch or YYYY/MM/DD or full "YYYY/MM/DD HH:MM:SS" format)
start	When eventlog is a database (db, odbc), it is the start recnum record. The recnum is returned by previous calls or can be any known value. For a flatfile eventlog, this filter returns the events after 'start' number of events.
end	The end recnum record. The recnum is returned by previous calls or can be any known value.
offset	When the eventlog is a database, the filter offset retrieves the events after 'offset' number of events. When the eventlog is a flat file, offset is the position in the file and should be the end of an event. Offsets for a flatfile can be known from previous calls. It is optional, but greatly speeds up retrieval, if provided.
verbose=0 1	By default, only the common attributes are returned. However, if verbose is set (verbose=1) the complete event is returned. Default=0
limit	Limit on records returned. The default is set to 32768, or 512 if verbose is set, to stop massive data results being returned by default.
order	Attribute to retrieve the data in. For example, epoch to get records in date order, or recnum to retrieve in logged order.  Note: Applicable only when eventlog is a database.
orderdirection=asc desc	Attribute specifying "asc" or "desc" ordering.  Note: Applicable only when eventlog is a database.
dump=0 1	When set (dump=1), retrieves accept and finish events separately. Default value: dump=0  Note: Filter available from PMUL version 21.1
uniqueid, etype, runhost, submithost, runuser, submituser, runcommand, exitstatus	etype=[A R K] for 'A'ccept, 'R'eject and 'K'eystroke events respectively. [runhost submithost runuser submituser runcommand exitstatus] = <value> These are the wildcard matched attributes to filter the events based on a value specified. Example: <pre>pbrstcall -a <appid> -k <key> -l -X GET https://<host>:24351/REST/v2/events etype=[A R K] runhost submithost runuser submituser runcommand exitstatus]=<value></pre>

Finish events can be retrieved only when dump=1

```
pbrestcall -a <appid> -k <key> -l -X GET
https://<host>:24351/REST/v2/events etype=F dump=1
runhost|submithost|runuser|submituser|runcommand|exitstatus]=<value>
```

recnum is returned for every retrieved record and can be used as the next start. Flat-file searching also returns offset which can be used in next requests.

To fetch events in batches, use offset attribute for database type eventlog and use start attribute for a flatfile.

If you know the file offset of previous record for a flatfile eventlog, you can mention offset=<fileoffset> to get next set of events in flatfile type of eventlog, otherwise use start=<count of events> to fetch the events in batches. File offset for a flatfile eventlog can be known from previous REST calls.

Example:

In an eventlog database, to fetch the 3rd batch of events with a batchsize of 100, use the API as below.

```
pbrestcall -a <appid> -k <key> -l -X GET https://<host>:24351/REST/v2/events offset=200 limit=100
```

In a flatfile eventlog, to fetch the 3rd batch of events with the batch size of 100, use the API as below.

```
pbrestcall -a <appid> -k <key> -l -X GET https://<host>:24351/REST/v2/events start=200 limit=100
```

Following is the syntax of a REST call with all the attributes and filters,

```
pbrestcall -a <appid> -k <key> -l -X GET
https://<host>:24351/REST/v2/events order=<field> orderdirection=[asc|desc] limit=<number_of_
rows> start=<start_recnum> end=<end_recnum> offset=<offset> from=<from_date> to=<to_date>
verbose=[1|0] dump=[1|0] etype=[A|R|K|F]
[runhost|submithost|runuser|submituser|runcommand|exitstatus]=<*>
```

Other attributes include format, file and dsn. Following section describes more about them.

List Events by Eventdestination

For SQLite DB

```
GET https://<host>:24351/REST/v2/events?appid=<appid>&timestamp=<timestamp>&hmac=<hmac>&format=db
```

```
pbrestcall -a <appid> -k <key> -l -X GET https://<host>:24351/REST/v2/events format=db
```

or nothing since that is the default.

To retrieve events from a specific SQLite database file,

```
GET
```

```
https://<host>:24351/REST/v2/events?appid=<appid>&timestamp=<timestamp>&hmac=<hmac>&format=db&file=<file>
```

```
pbrestcall -a <appid> -k <key> -l -X GET https://<host>:24351/REST/v2/events format=db file=<file>
```

For ODBC

You must set the DSN.

```
GET https://<host>:24351/REST/v2/events?appid=<appid>&timestamp=<timestamp>&hmac=<hmac>&format=odbc&dsn=<DSN>
```

```
pbrestcall -a <appid> -k <key> -l -X GET https://<host>:24351/REST/v2/events format=odbc dsn=<DSN>
```

For Flat File

You must set the file name.

```
GET https://<host>:24351/REST/v2/events?appid=<appid>&timestamp=<timestamp>&hmac=<hmac>&format=ff&file=<file>
```

```
pbrestcall -a <appid> -k <key> -l -X GET https://<host>:24351/REST/v2/events format=ff file=<file>
```

You can also use an old version of API with limited filters for a flat file.

```
GET https://pbuild:24351/REST/events?appid=<appid>&timestamp=<timestamp>&hmac=<hmac>
```

```
RESPONSE {"time":0,"events":
[{"runuser":"root","id":"c0a8108a52775f9781521","time":"08:49:27","masterhost":"pbuild","submithost":"pbuild","event":"Accept","argv":
["typeset","x","SHELL","PATH","HOME"],"runhost":"pbuild","date":"2013/11/04","user":"root","exitstatus":"local
shellbuiltin","runargv":["typeset","-x","SHELL","PATH","HOME"]},
{"runuser":"root","id":"c0a8108a528ce1a0462F","time":"16:21:52","masterhost":"pbuild","submithost":"pbuild","event":"Reject","argv":
["bash"],"runhost":"pbuild","date":"2013/11/20","user":"root","exitstatus":"","runargv":
["bash"]},
{"runuser":"root","id":"c0a8108a5285f5d350FE","time":"10:22:11","masterhost":"pbuild","submithost":"pbuild","event":"Accept","argv":
["pbguid","policy"],"runhost":"pbuild","date":"2013/11/15","user":"root","exitstatus":"Authorized","runargv":["pbguid","policy"]},
```



```
{"runuser":"ctaylor","id":"c0a8108a528ce3414793","time":"16:28:49","masterhost":"pbuild","submithost":"pbuild","event":"Accept","argv":["CSV","ctaylor","udev","bash#csh"],"runhost":"192.168.16.138","date":"2013/11/20","user":"ctaylor","exitstatus":"Command finished with exit status 0","runargv":["echo",""]}, ...
```

To enable the filtering of events, parameters can be passed to this REST endpoint.

<code>accept=0 1</code>	(default=1) Return Accept events
<code>reject=0 1</code>	(default=1) Return Reject events
<code>keystroke=0 1</code>	(default=1) Return events that resulted in an IO Log

These three filters are combined logically "OR" and should all be specified to limit the return of events.

File Integrity Monitoring (FIM)

FIM - Get Config

Retrieve the specified FIM configuration policy.

```
GET https://server1:24351/REST/fim/configs?appid=<appid>&timestamp=<timestamp>&hmac=<hmac>
```

```
RESPONSE {"name":"default","cfg":{"predefs":{"bin":{"all":{"ino":true,"mode":true,"uid":true,"gid":true,"own":"root","pmask":"022","size":true,"mtime":true,"ctime":true,"hash":true,"risk":10},"sysconf":{"exec":{"ino":true,"mode":true,"uid":true,"gid":true,"pmask":"022","size":true,"mtime":true,"ctime":true,"hash":true,"risk":10},"script":{"ino":true,"mode":true,"uid":true,"gid":true,"pmask":"022","size":true,"mtime":true,"ctime":true,"hash":true,"risk":10},"dev":{"uid":true,"gid":true,"mode":true,"risk":10},"other":{"ino":true,"mode":true,"uid":true,"gid":true,"pmask":"002","size":true,"mtime":true,"ctime":true,"risk":6},"log":{"all":{"uid":true,"gid":true,"mode":true,"pmask":"002","mtime_later":true,"ctime_later":true,"risk":4}}},"include":[{"path":"/etc/*","chk":"sysconf","recurse":true,"xdev":true,"follow":false}, {"path":"/proc","chk":"log","recurse":false}, {"path":"/mnt","chk":"log","recurse":false}, {"path":"/etc/mstab","chk":"log","recurse":false}, {"path":"/etc/motd","chk":"log","recurse":false}, {"path":"/etc/passwd","chk":"log","recurse":false}, {"path":"/etc/shadow","chk":"log","recurse":false}, {"path":"/boot/*","chk":"sysconf","recurse":true,"xdev":true,"follow":false}, {"path":"/bin/*","chk":"bin","recurse":true,"xdev":true,"follow":false}, {"path":"/var/log/*","chk":"log","recurse":true,"xdev":true,"follow":false}, {"path":"/var/adm/*","chk":"log","recurse":true,"xdev":true,"follow":false}], "exclude":["/root/.sh_history","/home/*","/etc/pb.db"]}}
```

FIM - List Configuration Assignments

List all of the FIM configuration policies.

```
GET https://server1:24351/REST/fim/assign?appid=<appid>&timestamp=<timestamp>&hmac=<hmac>
```

```
RESPONSE {"name":"default","hostname":"pbuild","lastupdated":"2016-11-14 16:17:08"}
```

FIM - Get configuration Assignment for Host

Get the name of the currently assigned configuration policy.

```
GET https://server1:24351/REST/fim/config/host?appid=<appid>&timestamp=<timestamp>&hmac=<hmac>&hostname=<hostname>
```

```
GET https://server1:24351/REST/fim/config/host?appid=<appid>&timestamp=<timestamp>
&hmac=<hmac>&name=<name>
```

```
RESPONSE {"name":"default","hostname":"pbuild","lastupdated":"2016-11-14 16:17:08"}
```

FIM - List Reports

List the FIM reports.

```
GET https://server1:24351/REST/fim/reports?appid=<appid>&timestamp=<timestamp>&hmac=<hmac>
```

```
RESPONSE [{"uuid":"d897e8d5-d854-450a-be83-faeef2d52dae","rundate":"2016-11-14
19:43:41","updated":0,"deleted":false,"new":3332,"total":3332,"policy":0,"max_
risk":10,"name":"default","host":"pctest"}]
```

FIM - Get Report

Retrieve the specified report.

```
GET https://server1:24351/REST/fim/report?appid=<appid>&timestamp=<timestamp>
&hmac=<hmac>&uuid=<uuid>
```

```
RESPONSE {"uuid":"8511cd6f-f21e-4832-ba2a-
33b2c483012f","rundate":26860624,"updated":1,"deleted":0,"new":0,"total":1,"policy":0,"max_
risk":6,"name":"default","host":"pbuild","rpt":[{"after":{"mtime":"2016-11-15
13:20:16","ctime":"2016-11-15
13:20:16","ino":544386,"dev":2051,"file":"foo"},"path":"/etc","before":{"mtime":"2016-11-04
16:14:03","ctime":"2016-11-04
16:14:03","ino":544386,"dev":2051,"file":"foo"},"risk":6,"change":"updated"}]}
```

FIM - Put Configuration

Put the specified configuration policy.

```
PUT https://server1:24351/REST/fim/config?appid=<appid>&timestamp=<timestamp>&hmac=<hmac>
```

```
REQUEST { JSON config }
```

```
RESPONSE { "status" : 0 }
```

FIM - Assign Configuration

Assign the specified host to the named configuration policy.

```
PUT https://server1:24351/REST/fim/assign?appid=<appid>&timestamp=<timestamp>&hmac=<hmac>
```

```
REQUEST { "name" : "<configname>", "hostname" : "<hostname>" }
```

```
RESPONSE { "status" : 0 }
```

FIM - Delete Config

Delete the specified policy configuration.

```
DELETE https://server1:24351/REST/fim/config?appid=<appid>&timestamp=<timestamp>  
&hmac=<hmac>&name=<name>
```

FIM - Delete Report

Delete the specified FIM report.

```
DELETE https://server1:24351/REST/fim/report?appid=<appid>&timestamp=<timestamp>&hmac=<hmac>
```

Key File

Key Get

Gets the specified **pb.key** file as a base64 encoded string.

```
GET https://pbuild:24351/REST/key?appid=<appid>&timestamp=<timestamp>&hmac=<hmac>
```

```
RESPONSE {"keyfile":
{"blob": "A8EumiklyZHh2yhJ/VeUXI1117r7WlXKuOkr/MtHY2fjqTB8q0/h0/rpsoJZrRJGmwvYiODH/uLIpfh2srZ01meA
plTlYpUCWzLOp4Wmnfu4MWI7GtLYQnQbFjhm0jI3mrY8IDo0IusULmXTh7GCjYtS78Ypj3S2RZYQ2WNWZtoKtC2/Psv
+svTU9/Sj7K+Wna6zFvTnwqVTkdy8rOQ+kGDoBrZV0QbVowD2jWnGQjNjdne/w4mSdnX5BMATCe93laKbr4c+6MV
97S/HW9HUu/bxG7pf+XcovaX6d7WPRJYGVQP/GLJ6LER34MRpsuGJf3dMA4jvRGTPKue685pi41FyvIhacPKWBHBcD
e3mmIDlfoU7zzJy3k5hG1OEYARoAC/iBSjLQqIv4pTJr1FG95ko8T/DsVigg1VuFiB1VZg45FODzQy2sm5FHu47828kthvO
n30HXZBFFug2z7qcmohRGnKHHh9QSRzzy0GoaDs/+DTKljOatwHXkzKBGLdyKRO068kk6xN9EafHWxq69zJ8v5nBZRL
8aOKn/4341ULLGsmPEtf+bNxxJ7Wtlfw3pt3FSidC3ikButt3giM1dD42qYp2YYa19U7kNxC1+UiYK7EUJD7JI26MM/WiA3
uWctkrz6iSnMxIZhhpYxV40jPFBNJg3entziObCw8DVzkthPmYA+biXh05gnpUV7pSvq0XrefJ6/dXWLNosSoQD/znsFab
GzX/trWu1CNVqt1CW8nPzJD6gNFieWMavnipolDza+9Gdo9Sw1tFZyRip/ZZ56jvhnCKmOS8xOhfXx2c05kXwpBoBto8
FCrkiwjCi3f3YDz/RrNCSpvWSgJN88MnNCXJRWy+cyQu7c/MQcwu+ySK1g0tqUhTLgKCj7RwnGXrsUBu0F4WI9q6VCp
vfXDvnGCwsEwoyEnKMsmKGR30q6fM0tN+npq0zFLvnaAjtIqg0AxP4aHe6b6PLR/efBt92EoyY6TbN+FsthUyjRTSbwh
lVnGs/29lWemUJ4Pjss5FVBKoX5dKptivUDyvjvChG7RQl/UuUzPtVvDr1C/oGPOf8VBT1j1gl9s9w6GioofgoxW+/tGISwC
zQYgKkvNrAOLTTIGS2x/1WAxCZFCZ74bfp/bsopnDxD+QzMrIBRQf0x9HdXC6vZPJvOdYIBCACp2fcwFE5aZ8y9o+gH3
cia2L0G5r07xEvIFp64xLKpBwCR0yJXg0uwjIrlaWHk7BVUKUDeISH1KSyuZQwMskgaGtIZcPyyTmSnxYKIGHQnPH54F4
4/4FLeDwnfiMGABN3WjOFvg9PvKcTJsa/0OUF4yw/oW7vGKaiJqG7bZ0dsS0FRW/nQ3XSVUQ5p7oFX7", "blen":1026}
,"status":0}
```

Key Set

Sets the specified **pb.key** to the base64 encoded string.

```
PUT https://pbuild:24351/REST/key?appid=<appid>&timestamp=<timestamp>
&hmac=<hmac>&file=%2Fetc%2Fpb%2Ftmp.key
```

```
REQUEST
{"blob": "WFP8yHAhy2KSub1I34/azYmZeDHi3V1C+9H1nlmCthxqFD3B7ByQV8V3dCYP5N6S2nbIIHOWQ+emfzi5BS
YUxwRUqks7dy5FnaquMVyIVF1lehyqDufH38n33XOyN6dJJptcQS9bmlyoGT3IUrkyG1U6sHAN2k3HIDK4xDwu+wpgn
I8lLOQni6A2n4Hg7FHUcuvhtQG1D0XThe7qs1bEx4gb3o8ONR+dUMvvDpe40Sr6LCV8Lj3C8ixN1c8eqlMyxnUZr266J
5otWz+XCj44BhDndikJqwIucNADo50jnZL6leUmfds+5xoFip1UmrBzsdhpYf9H/Uo5zmICMJjko+4VDAyTBhCmbEH63
MaQ6kXef3NstyUa4js7Xw6nzBj2n2aXkQd0HFccSalpMMpFdB8v9VBgdKpAATq51bAwqREfO+LbEfNLTWvtKs9X/F36

3rI8dYCIyHqfz62sSFEqDcFuXKkp1HTagZL9tYk55LtS+mbFiDq7VvnpTBGaIL29Y+G3sSt/Stz2sUh7Fpjy3D30IgpmyNjHn
j
ehmULOrjm6Rv1aFvNILzBLKV63HhgMzYJ+FzcV9X9HEHWXAdfwrYnX+KGPqmyD8Prd6RSIX9gJUBks0VPfdmoEshS
W5iCXoStyPPGNQ1fFECXQK69Pdrqr3LTm3L2Lu0uQnNTBTfvdull+goGKiAffJAY6kLOH5UTyC1CwZzkLI5Xb8EJ8nPT1
Ru5VBXI3f0XHf97UFIKeCLzC2ewW1sv09NFqrTQG0Ie6uEsXJoSul2/196bB5A50zd8DVgnt57PO8g09Y7hYkItYFFPNk5
UtAL+86Bx1ZFTbyiooCwHwrZbp9hBjAYhz84U7eKuPxFkZL/Z9A3lwFmvdibPFwL+I2KWOED9b90wmQsC+RJZ7rm2x
JVF501D0ETm2Ot7vXIy5Kka2UakLJPUpQtLPAVAnbwBRHZ71j+D9UAczWth1YUmRPcKNNnvRfj+grtvBb63oOouCUg
ZxPXSJfJ+b7Hwghi84ZTbxrSAZt2VRgUx01uSP0aogT9fllH7fuMjCtGPMsvBFgg/QvEwz10PEeY1DY8W5zVal1QMxf1bBQ
JOQ+7iZLwG/pWnxo04tBYE5mxbt3AL9KHK9AIMx8e/vqSeu7cxJo2PAjnkEp+R1Q6gjjpRCD19f0zQNV4IhRFFJ3eItF7
```

```
p0O3WS1iCPNATmpu+ZpYYeChCJltL3+W8Tk8AsLCMTweZoJugFBKZ8fQR6WlPG6mzD98jgbet8czjpGpjzA41", "blen":1026}
```

```
RESPONSE {"status":0}
```

Key New

Creates a new specified **pb.key** file and generates random contents.

```
POST https://pbuild:24351/REST/key?appid=<appid>&timestamp=<timestamp>&hmac=<hmac>&file=%2Fetc%2Fpb%2Ftmp.key
```

```
REQUEST {}
```

```
RESPONSE {"status":0}
```

Get Key file as attachment

Retrieves the specified **pb.key** file as a binary attachment.

```
GET https://pbuild:24351/REST/keyfile?appid=<appid>&timestamp=<timestamp>&hmac=<hmac>&file=%2Fetc%2Fpb.key
```

Registry Name Service

Registry Name Service

Registry Name Service Cache Update

Retrieves all the Registry Name Service Database updates for this host since **lastupdated**.

```
GET https://server1:24351/REST/service/svccache?appid=<appid>&timestamp=<timestamp>
&hmac=<hmac>&uuid=<uuid>&lastupdated=<num>
```

```
RESPONSE {"services":[{"svcgname":"registry_name_service","svc":1,"cn":"pbuild","uuid":"7d4504dd-
b64f-453c-9a12-53a079a8d4ff","fqdn":"pbuild","addrs":
[{"family":4,"addr":"192.168.16.138","port":24351}], "role":4,"sorder":1,"lastupdated_
usec":1479140226550825}, {"svcgname":"dflt_pbpolicy_
service","svc":2,"cn":"pbuild","uuid":"7d4504dd-b64f-453c-9a12-
53a079a8d4ff","fqdn":"pbuild","addrs":
[{"family":4,"addr":"192.168.16.138","port":24351}], "role":4,"sorder":1,"lastupdated_
usec":1479140226572396}, {"svcgname":"dflt_log_service","svc":4,"cn":"pbuild","uuid":"7d4504dd-
b64f-453c-9a12-53a079a8d4ff","fqdn":"pbuild","addrs":
[{"family":4,"addr":"192.168.16.138","port":24351}], "role":4,"sorder":1,"lastupdated_
usec":1479140226597899}, {"svcgname":"dflt_sudopolicy_
service","svc":8,"cn":"pbuild","uuid":"7d4504dd-b64f-453c-9a12-
53a079a8d4ff","fqdn":"pbuild","addrs":
[{"family":4,"addr":"192.168.16.138","port":24351}], "role":4,"sorder":1,"lastupdated_
usec":1479140226627291}, {"svcgname":"dflt_fim_service","svc":128,"cn":"pbuild","uuid":"7d4504dd-
b64f-453c-9a12-53a079a8d4ff","fqdn":"pbuild","addrs":
[{"family":4,"addr":"192.168.16.138","port":24351}], "role":4,"sorder":1,"lastupdated_
usec":1479140226674180}, {"svcgname":"dflt_logarch_
service","svc":32,"cn":"pbuild","uuid":"7d4504dd-b64f-453c-9a12-
53a079a8d4ff","fqdn":"pbuild","addrs":
[{"family":4,"addr":"192.168.16.138","port":24351}], "role":4,"sorder":1,"lastupdated_
usec":1479140226701934}, {"svcgname":"dflt_pbpolicy_
service","svc":2,"cn":"pbtest","uuid":"024352a4-d4d0-48d2-bdcc-
76ec429632f7","fqdn":"pbtest","addrs":
[{"family":4,"port":24351,"addr":"192.168.16.184"}], "role":1,"sorder":0,"lastupdated_
usec":1479152447772617}, {"svcgname":"dflt_log_service","svc":4,"cn":"pbtest","uuid":"024352a4-
d4d0-48d2-bdcc-76ec429632f7","fqdn":"pbtest","addrs":
[{"family":4,"port":24351,"addr":"192.168.16.184"}], "role":1,"sorder":0,"lastupdated_
usec":1479152447774991}, {"svcgname":"dflt_fim_service","svc":128,"cn":"pbtest","uuid":"024352a4-
d4d0-48d2-bdcc-76ec429632f7","fqdn":"pbtest","addrs":
[{"family":4,"port":24351,"addr":"192.168.16.184"}], "role":2,"sorder":2,"lastupdated_
usec":1479152505459492}]}
```

Registry Name Service - Get Service Group Info

Retrieves Service Group information from the Registry Name Service Database.

```
GET https://server1:24351/REST/service/svcgrp?appid=<appid>&timestamp=<timestamp>&hmac=<hmac>
```

```
RESPONSE [{"svcgid":1,"svcgname":"registry_name_service","svc":"registry","updated_usec":"2016-11-14 16:17:06","deleted":false},
{"svcgid":2,"svcgname":"dflt_pbpolicy_service","svc":"pbpolicy","updated_usec":"2016-11-14 16:17:06","deleted":false},
{"svcgid":3,"svcgname":"dflt_log_service","svc":"logsvr","updated_usec":"2016-11-14 16:17:06","deleted":false},
{"svcgid":4,"svcgname":"dflt_sudopolicy_service","svc":"sudopolicy","updated_usec":"2016-11-14 16:17:06","deleted":false},
{"svcgid":5,"svcgname":"dflt_Solr_service","svc":"Solr","updated_usec":"2016-11-14 16:17:06","deleted":false},
{"svcgid":6,"svcgname":"dflt_logarch_service","svc":"logarchive","updated_usec":"2016-11-14 16:17:06","deleted":false},
{"svcgid":7,"svcgname":"dflt_beyondinsight_service","svc":"beyondinsight","updated_usec":"2016-11-14 16:17:06","deleted":false},
{"svcgid":8,"svcgname":"dflt_fim_service","svc":"fim","updated_usec":"2016-11-14 16:17:06","deleted":false}]
```

Registry Name Service - Get Registry Name Service Host and Service Group Info

Retrieves Registry Name Service Host and Services information.

```
GET https://server1:24351/REST/service/svchost/name?appid=<appid>&timestamp=<timestamp>
&hmac=<hmac>&cn=<cn>&svcgname=<svcgname>
```

```
RESPONSE [{"svcgid":1,"svcgname":"registry_name_service","svc":"registry","updated_usec":"2016-11-14 16:17:06","deleted":false,"svcs":
[{"svcgid":1,"hostid":1,"role":"primary","sorder":1,"created_usec":"2016-11-14 16:17:06","updated_usec":"2016-11-14 16:17:06","cn":"pbuild","uuid":"7d4504dd-b64f-453c-9a12-53a079a8d4ff","fqdn":"pbuild","adrs":
[{"family":4,"addr":"192.168.16.138","port":24351}], "tnlzone":0,"deleted":0}],
{"svcgid":2,"svcgname":"dflt_pbpolicy_service","svc":"pbpolicy","updated_usec":"2016-11-14 16:17:06","deleted":false,"svcs":[{"svcgid":2,"hostid":1,"role":"primary","sorder":1,"created_
usec":"2016-11-14 16:17:06","updated_usec":"2016-11-14 16:17:06","cn":"pbuild","uuid":"7d4504dd-b64f-453c-9a12-53a079a8d4ff","fqdn":"pbuild","adrs":
[{"family":4,"addr":"192.168.16.138","port":24351}], "tnlzone":0,"deleted":0}],
{"svcgid":3,"svcgname":"dflt_log_service","svc":"logsvr","updated_usec":"2016-11-14 16:17:06","deleted":false,"svcs":[{"svcgid":3,"hostid":1,"role":"primary","sorder":1,"created_
usec":"2016-11-14 16:17:06","updated_usec":"2016-11-14 16:17:06","cn":"pbuild","uuid":"7d4504dd-b64f-453c-9a12-53a079a8d4ff","fqdn":"pbuild","adrs":
[{"family":4,"addr":"192.168.16.138","port":24351}], "tnlzone":0,"deleted":0}],
{"svcgid":4,"svcgname":"dflt_sudopolicy_service","svc":"sudopolicy","updated_usec":"2016-11-14 16:17:06","deleted":false,"svcs":[{"svcgid":4,"hostid":1,"role":"primary","sorder":1,"created_
usec":"2016-11-14 16:17:06","updated_usec":"2016-11-14 16:17:06","cn":"pbuild","uuid":"7d4504dd-b64f-453c-9a12-53a079a8d4ff","fqdn":"pbuild","adrs":
[{"family":4,"addr":"192.168.16.138","port":24351}], "tnlzone":0,"deleted":0}],
{"svcgid":5,"svcgname":"dflt_Solr_service","svc":"Solr","updated_usec":"2016-11-14 16:17:06","deleted":false},
{"svcgid":6,"svcgname":"dflt_logarch_service","svc":"logarchive","updated_usec":"2016-11-14 16:17:06","deleted":false,"svcs":[{"svcgid":6,"hostid":1,"role":"primary","sorder":1,"created_
usec":"2016-11-14 16:17:06","updated_usec":"2016-11-14 16:17:06","cn":"pbuild","uuid":"7d4504dd-b64f-453c-9a12-53a079a8d4ff","fqdn":"pbuild","adrs":
[{"family":4,"addr":"192.168.16.138","port":24351}], "tnlzone":0,"deleted":0}],
{"svcgid":7,"svcgname":"dflt_beyondinsight_service","svc":"beyondinsight","updated_usec":"2016-
```



```
11-14 16:17:06", "deleted": false},
{"svcgid": 8, "svcgname": "dflt_fim_service", "svc": "fim", "updated_usec": "2016-11-14
16:17:06", "deleted": false, "svcs": [{"svcgid": 8, "hostid": 1, "role": "primary", "sorder": 1, "created_
usec": "2016-11-14 16:17:06", "updated_usec": "2016-11-14 16:17:06", "cn": "pbuild", "uuid": "7d4504dd-
b64f-453c-9a12-53a079a8d4ff", "fqdn": "pbuild", "adrs":
[{"family": 4, "addr": "192.168.16.138", "port": 24351}], "tnlzone": 0, "deleted": 0},
{"svcgid": 8, "hostid": 2, "role": "secondary", "sorder": 2, "created_usec": "2016-11-14
19:41:45", "updated_usec": "2016-11-14 19:40:47", "cn": "pbtest", "uuid": "024352a4-d4d0-48d2-bdcc-
76ec429632f7", "fqdn": "pbtest", "adrs":
[{"family": 4, "port": 24351, "addr": "192.168.16.184"}], "tnlzone": -1, "deleted": 0}}}]
```

Registry Name Service - Get Registry Name Service Group and Role Information

Retrieves Registry Name Service Group and Role information.

```
GET https://server1:24351/REST/service/svchost/role?appid=<appid>&timestamp=<timestamp>
&hmac=<hmac>&role=<role>&svcgname=<svcgname>
```

```
RESPONSE {"svcgid": 1, "svcgname": "registry_name_service", "svc": "registry", "updated_usec": "2016-11-
14 16:17:06", "deleted": false, "svcs": [{"svcgid": 1, "hostid": 1, "role": "primary", "sorder": 1, "created_
usec": "2016-11-14 16:17:06", "updated_usec": "2016-11-14 16:17:06", "cn": "pbuild", "uuid": "7d4504dd-
b64f-453c-9a12-53a079a8d4ff", "fqdn": "pbuild", "adrs":
[{"family": 4, "addr": "192.168.16.138", "port": 24351}], "tnlzone": 0, "deleted": 0}}}]
```

Registry Name Service - Get Registry Name Service Host information

Retrieves Registry Name Service Group and Role information.

```
GET https://server1:24351/REST/service/host/name?appid=<appid>&timestamp=<timestamp>
&hmac=<hmac>&cn=<cn>
```

```
GET https://server1:24351/REST/service/host/uuid?appid=<appid>&timestamp=<timestamp>
&hmac=<hmac>&uuid=<uuid>
```

```
GET https://server1:24351/REST/service/host/fqdn?appid=<appid>&timestamp=<timestamp>
&hmac=<hmac>&fqdn=<fqdn>
```

```
RESPONSE {"hostid": 1, "cn": "pbuild", "uuid": "7d4504dd-b64f-453c-9a12-
53a079a8d4ff", "fqdn": "pbuild", "adrs":
[{"family": 4, "addr": "192.168.16.138", "port": 24351}], "tnlzone": 0, "updated_usec": "2016-11-14
16:17:06", "deleted": false}
```

Registry Name Service - Delete Service Group

Delete the specified Service Group.

```
DELETE https://server1:24351/REST/service/svcgrp?appid=<appid>&timestamp=<timestamp>
&hmac=<hmac>&svcgname=<svcgname>
```

Registry Name Service - Delete Service Host

Delete the specified host from the specified Service Group.

```
DELETE https://server1:24351/REST/service/svchost?appid=<appid>&timestamp=<timestamp>
&hmac=<hmac>&svcgname=<svcgname>&cn=<cn>
```

Registry Name Service - Delete Host

Delete the specified host from the Registry Name Service.

```
DELETE
https://server1:24351/REST/service/host?appid=<appid>&timestamp=<timestamp>&hmac=<hmac>&cn=<cn>
```

Registry Name Service - Put Service Group

Add/Update the specified Service Group.

```
PUT https://server1:24351/REST/service/svcgrp?appid=<appid>&timestamp=<timestamp>&hmac=<hmac>
```

```
REQUEST { "svc" : { "svcgname" : "<svcgname>", "svc" : "<svc>" }}
```

```
RESPONSE { "status" : 0 }
```

Registry Name Service - Put Service Group Host

Assign the specified host as a given role within the specified Service Group.

```
PUT https://server1:24351/REST/service/svchost?appid=<appid>&timestamp=<timestamp>&hmac=<hmac>
```

```
REQUEST { "svc" : { "svcgname" : "<svcgname>", "cn" : "<cn>", "role" : "<role>" }}
```

```
RESPONSE { "status" : 0 }
```

Registry Name Service - Put Host

Add/Update the named host in Registry Name Service.

```
PUT https://server1:24351/REST/service/host?appid=<appid>&timestamp=<timestamp>&hmac=<hmac>
```

```
REQUEST { "svc" : { "cn" : "<cn>", "fqdn" : "<fqdn>", "uuid", "<uuid>" } }
```

```
RESPONSE { "status" : 0 }
```

Registry Name Service - Promote

Promote the specified server to primary with its service group.

```
PUT https://server1:24351/REST/service/promote?appid=<appid>&timestamp=<timestamp>&hmac=<hmac>
```

```
REQUEST {"svcgname" : "<svcgname>", "cn" : "<cn>" }
```

```
RESPONSE { "status" : 0 }
```

Sync

REST call to set a file to be synchronized within a service group. This is the same as **pbadmin -cfg -A <file> <svgroup>**.

Set to Sync

```
pbrestcall -a <appid> -k <appkey> -l -X PUT https://<host>:<restport>/REST/v1/cfg/sync -d ' {"file":"<file>", "svcs":["<servicegroup>"]}'
```



Example:

```
pbrestcall -a admin -k d39cf6ea-e292-4c59-b60a-2d9650eb9cad -l -X PUT https://localhost:29106/REST/v1/cfg/sync -d '{"file":"/opt/pbul/policies/myfile", "svcs":["dflt_pbpolicy_service"]}'
```

Import

```
pbrestcall -a <appid> -k <appkey> -l -X PUT https://<host>:<restport>/REST/v1/cfg/import -d ' {"file":"<filename>"}
```



Example:

```
pbrestcall -a admin -k d39cf6ea-e292-4c59-b60a-2d9650eb9cad -l -X PUT https://localhost:29106/REST/v1/cfg/import -d '{"file":"/opt/rnsbul/policies/myfile"}
```

If file already exists, use **force**:

```
pbrestcall -a <appid> -k <appkey> -l -X PUT https://<host>:<restport>/REST/v1/cfg/import -d '{  
"force":1, "file":"<filename>"}'
```

Role Based Policy Database Manipulation

The functionality developed to manipulate the Role Based Policy Database was written with both the Command Line utility (**pbdbutil**), the policy server (**pbmasterd**) and the REST interface in mind. Functions use JSON objects to specify records to retrieve, update and delete.

To retrieve the entire Role Based Policy database, use the **REST GET HTTP** method, and specify a URL similar to **https:// ... /policies/rbp**.

To retrieve individual Role Based Policy entities, use the **REST GET HTTP** method with a URL similar to **https:// ... /policy/rbp/<entity>** and specify a parameter of either **name=<name>** or **id=<id>**.



Example:

```
https:// ... /policy/rbp/usergrp?.....name=ugrp1
```

To import a new Role Based Policy database, use the **REST PUT HTTP** method with a URL similar to **https:// ... /policies/rbp** and specify the complete database, in the appropriate format, in the **BODY** data.

To update specific Role Based Policy entities, use the **REST PUT HTTP** method with a URL similar to **https:// ... /policy/rbp/<entity>** and specify the entity, in the appropriate format, in the **BODY** data.

To delete specific Role Based Policy entities, the developer should use the **REST DELETE HTTP** method with a URL similar to **https:// ... /policy/rbp/<entity>** and specify a parameter of either **name=<name>** or **id=<id>**.

To begin a Role Based Policy Change Transaction, use the **REST PUT HTTP** method with a URL similar to **https:// ... /policy/rbp/begin**.

To commit a Role Based Policy Change Transaction, use the **REST PUT HTTP** method with a URL similar to **https:// ... /policy/rbp/commit**.

To rollback a Role Based Policy Change Transaction, use the **REST PUT HTTP** method with a URL similar to **https:// ... /policy/rbp/rollback**.

To retrieve a Role Based Policy Change Transaction details, use the **REST GET HTTP** method with a URL similar to **https:// ... /policy/rbp/transaction**.

Role Based Policy - Miscellaneous Calls

Retrieve RBP Version List

```
GET https://pbuild:24351/pbrest/REST/v2.0/policy/rbp/list?appid=<appid>&timestamp=<timestamp>
&hmac=<hmac>
```

```
RESPONSE { "rbp": [{ "version": 1, "who": "ctaylor", "why": "New data",
"created": 1529322345} ] }
```

Retrieve Entitlement "Raw" Data

```
GET https://pbuild:24351/pbrest/REST/v2.0/policy/rbp/entitlement?appid=<appid>
&timestamp=<timestamp>&hmac=<hmac>
```

Optional Arguments

- **submituser=<wildcard>**
- **runuser=<wildcard>**
- **submithost=<wildcard>**
- **runhost=<wildcard>**
- **command=<wildcard>**

```
RESPONSE {
"results": [
{
"id": 1,
"name": "Admin",
>tag": null,
"description": "Super users and admins",
"rorder": 1,
"action": "allowed",
"iolog": true,
"auth": false,
"script": false,
"message": false,
"submitusers": {
"Admins": {
"description": "Admin users",
"list": [
"root",
"admin"
]
}
}
},
},
},
```

```
"submithosts": {
  "All Hosts": {
    "description": "All Hosts",
    "list": [
      "*"
    ]
  },
  "runusers": {
    "Admins": {
      "description": "Admin users",
      "list": [
        "root",
        "admin"
      ]
    },
    "Users": {
      "description": "Normal Users",
      "list": [
        "user*"
      ]
    }
  },
  "runhosts": {
    "All Hosts": {
      "description": "All Hosts",
      "list": [
        "*"
      ]
    }
  },
  "commands": {
    "User Commands": {
      "description": "Common UNIX Commands",
      "list": [
        { "cmd": "/bin/ls", "runcommand": "" },
        { "cmd": "/bin/ls *", "runcommand": "" },
        { "cmd": "/usr/bin/ls", "runcommand": "" },
        { "cmd": "/usr/bin/ls *", "runcommand": "" },
        { "cmd": "/bin/cat *", "runcommand": "" },
        { "cmd": "/usr/bin/cat *", "runcommand": "" }
      ]
    }
  },
  "time/dates": {
    "Any Time": {
      "description": "Any Time",
      "list": [
        {
          "dotw": {
            "mon": [
              15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15,
              15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15, 15 ]
            },
          "tue": [
```



```

Unix and Linux Role Based Policy Entitlement Report - Level 1\n-----
-----\nDate/Time: 2018-06-18 09:14:48\nUser: *\nBelongs to
the following Roles: \n
Admin,users\n=====
  1\nName:          Admin\nDescription:      Super users and admins\nAction:
allowed\nTag:          \nMembership:      Admins\n\nSubmit Host(s): Any PBUL Host\nRun Host(s):
Any PBUL Host\n\nCommands may be executed as user(s): root,admin,user*\n\nPlease use the '-u'
flag to select user at run time.\neg: pbrun -u runuser command [arguments]\n\nUser may request
the following commands using pbrun:\n/bin/find *,/usr/bin/ls,/bin/ls,/bin/cat *,/bin/ls
*,/usr/bin/ls *,/usr/bin/rm *,\n/usr/bin/cat *,/usr/bin/find *,/sbin/shutdown *,/bin/more
*,/bin/id,/usr/bin/more *,\n/usr/bin/mount *,/bin/lm *,/bin/mount *,/bin/rm *,/usr/sbin/shutdown
*,\n/usr/bin/lm *,/usr/bin/id,/sbin/ifconfig *,/usr/sbin/ifconfig
*\n\n\n=====
4\nName:          users\nDescription:      Normal users\nAction:          allowed\nTag:
\nMembership:      Users\n\nSubmit Host(s):
build.company.com,staging.company.com,nfs.company.com\nRun Host(s):
build.company.com,staging.company.com,nfs.company.com\n\nCommands will execute as user:
user*\n\nUser may request the following commands using pbrun:\n/usr/bin/ls,/bin/find
*,/bin/ls,/bin/cat *,/bin/ls *,/usr/bin/rm *,/usr/bin/ls *,\n/usr/bin/cat *,/usr/bin/find
*,/bin/id,/bin/more *,/usr/bin/more *,/bin/lm *,\n/bin/rm *,/usr/bin/lm *,/usr/bin/id\n\n\n"
}

```

Client Registration

Create Client Registration Profile

Create or update a client profile. The format of these profiles are detailed below.

```
PUT https://pbuild:24351/REST/register?appid=<appid>&timestamp=<timestamp>&hmac=<hmac>
```

```
RESPONSE { "status" : 0 }
```

Retrieve Client Registration Profile

Retrieve a client profile so that the client install can action the profile:

```
GET https://pbuild:24351/REST/register?appid=<appid>&timestamp=<timestamp>&hmac=<hmac>&profile=profile1
```

```
RESPONSE {"status": 0, "profile": [{"type": "settings", "fname": "/etc/pb.settings"}, {"sname": "networkencryption", "type": "save"}, {"sname": "restkeyencryption", "type": "save"}, {"sname": "sslservercertfile", "type": "save"}]}
```

Retrieve Client Registration Profile File Attachment

Retrieve a client profile so that the client install can action the profile:

```
GET https://pbuild:24351/REST/register/file?appid=<appid>&timestamp=<timestamp>&hmac=<hmac>&fname=%2fetc%2fpb%2esettings&index=0
```

RESPONSE

File attachment or **{"status": 8110}**

Retrieve List of Client Registration Profiles

Retrieve a list of client profiles that match the given profile wildcard:

```
GET https://pbuild:24351/REST/register/profiles?appid=<appid>&timestamp=<timestamp>&hmac=<hmac>&profile=prof%2A
```

```
RESPONSE {"status": 0, "profiles": [{"type": "settings", "fname": "/etc/pb.settings"}, {"sname": "networkencryption", "type": "save"}, {"sname": "restkeyencryption", "type": "save"}, {"sname": "sslservercertfile", "type": "save"}]}
```

Delete Client Registration Profile

Retrieve a client profile so that the client install can action the profile:

```
DELETE  
https://pbuild:24351/REST/register?appid=<appid>&timestamp=<timestamp>&hmac=<hmac>&profile=profile
```

```
RESPONSE { "status" : 0 }
```

SOLR

Solr Get

Retrieves SOLR search results based on the supplied criteria.

```
GET https://pbuild:24351/REST/Solr?appid=<appid>&timestamp=<timestamp>&hmac=<hmac>&query=%3A*
```

```
RESPONSE {"status":0,"iologs":[{"runuser":"root","endtime":"2013-12-05T15:20:18Z","id":"c0a8108a52a099b0586F","_version_":1453595633085579264,"starttime":"2013-12-05T15:20:16Z","replay_link":"http://pbuild:25348/iolog?file=/tmp/iolog.root.NMNzvX&foreground=White&background=Black&pause=1000&inputhistory=5&fontsize=10&submitButton=View+I/O+Log","runcommand":"bash","name":"/tmp/iolog.root.NMNzvX","runhost":"pbuild","user":"root","runargv":"bash"}, {"runuser":"root","endtime":"2013-12-05T14:52:13Z","id":"c0a8108a52a0931d5541","_version_":1453669846119088128,"starttime":"2013-12-05T14:52:13Z","replay_link":"http://pbuild:25348/iolog?file=/tmp/iolog.root.jVruz1&foreground=White&background=Black&pause=1000&inputhistory=5&fontsize=10&submitButton=View+I/O+Log","runcommand":"bash","name":"/tmp/iolog.root.jVruz1","runhost":"pbuild","user":"root","runargv":"bash"}, {"runuser":"root","endtime":"2013-12-05T14:53:02Z","id":"c0a8108a52a0934a558B","_version_":1453669852501770240,"starttime":"2013-12-05T14:52:58Z","replay_link":"http://pbuild:25348/iolog?file=/tmp/iolog.root.ckYxun&foreground=White&background=Black&pause=1000&inputhistory=5&fontsize=10&submitButton=View+I/O+Log","runcommand":"bash","name":"/tmp/iolog.root.ckYxun","runhost":"pbuild","user":"root","runargv":"bash"}]}
```

Elasticsearch Logstash API Calls

The PMUL REST API includes commands to manage credentials for Elasticsearch and Logstash implementations. Similar commands are also available with the pbdbutil CLI tool. privilege management for u

elkcred

Retrieves a single credential. Retrieve the credential associated with the passed ID. The password is not displayed.

GET

```
pbrestcall -X GET https://host:24351/REST/elkcred/id
```

PUT

Add or update a single credential. The JSON object passed in the **-d** argument must include, as a minimum, the ID and type fields. Other added fields depend on the credential type. The response indicates success or failure.

```
pbrestcall -X ... https://host:24351/REST/elkcred -d '{...}'
```

DELETE

Remove a single credential:

```
pbrestcall -X ... DELETE https://host:24351/REST/elkcred/id
```

The response indicates success or failure.

elkcreds

GET

Retrieve all credentials:

```
pbrestcall -X GET ... https://host:24351/REST/elkcreds
```

The response lists all credentials, including their ID values. The passwords are not displayed.

elkcredtest

GET

Test the connectivity of a credential that exists in the credential store:

```
pbrestcall -X GET ... https://host:24351/REST/elkcredtest/id
```

Tests the credential set in the URL (as ID) against all URLs configured in the **elkinstances** value in **/etc/pb.settings**. Other settings in that file (other than **elkcredential**) are applied as well.

Results are reported as an array of findings per URL in **elkinstances**. Because authenticating by token or API key can generate two HTTP requests, a test performed against a credential of one of these types probes two distinct URLs, as shown in the results given below:

```
[user@host]$ pbrestcall -l -X GET -a <app> -k <key> \
https://localhost:24351/REST/elkcredtest/elastic_token
{
  "results": [
    {
      "token-request": {
        "url": "https://elkhost:9200/_security/oauth2/token",
        "curlcode": "0 (No error)",
        "httpcode": "200 (OK)"
      },
      "test-request": {
        "url": "https://elkhost:9200/?pretty",
        "curlcode": "0 (No error)",
        "httpcode": "200 (OK)"
      }
    }
  ]
}
```

In this case, the data reported in the token-request object is the response to an attempt to acquire the token, and the result reported in the test-request object corresponds to the use of that token to authenticate against the URL specified in the test-request.



Note: Tests of token and apikey credentials against Logstash instances will fail.

POST

This can be used to test a credential that might not yet be in the credential store:

```
pbrestcall -X POST .. https://host:24351/REST/elkcredtest -d { .. }
```

The supplied POST data must be in the same format as specified for the three authentication types documented earlier in this guide. Using those data formats, a credential is tested against the URLs in the **elkinstances** value in **/etc/pb.settings**, and the test applies all other ELK-related settings from that file.

To test a credential completely independently of **/etc/pb.settings**, add an **elkinstances** element to the POST data:

```
{ .., "elkinstances": "elasticsearch=https://elkhost:9200" }
```



For more information on the authentication types, please see *"Credential Store"* on page 351.

Java Implementation of REST API

Full Java example sources are provided to allow developers to quickly interface with the Privilege Management for Unix and Linux REST API. The sources are developed with JDK 7 and Eclipse and an example project is provided in the **examples/java/PBULAPI** directory (use import project from the Eclipse menu). We recommend JUnit 4 be installed and configured to run the provided test suite.

- **org/json/*** contains the www.json.org simple java JSON implementation, and is included verbatim.
- **com/beyondtrust/pbul/*** contains all of the example code to call the REST API, including:
 - **PBULException**: An exception used throughout the whole package.
 - **PBULsession**: Keeps the session information to implement the REST API, including the host URI, application ID, and key. It is used by the objects below to setup and call the REST API.
 - **PBULutil**: Provides miscellaneous methods required by the implementation.
 - **PBULtype**, **PBULarray** and **PBULobject**: Types defined to provide the data types required to access the settings file.
 - **PBULevents**: Provides access to the event logs.
 - **PBULiologs**: Provides access to I/O logs, including listing, searching, and retrieving.
 - **PBULkey**: Provides access to retrieve and create **pb.key** files.
 - **PBULlicense**: Provides details of the licensing on the host.
 - **PBULpolicy**: Provides two main methods to access the policy files, whether they are normal pb scripts, or CSV format.
 - **PBULsolr**: Provides a front-end to the SOLR option provided in Privilege Management for Unix and Linux 8.
- The **test** folder provides a complete test suite containing test cases that call each of the provided methods and can also be used as examples.

X Window System Session Capturing

Introduction

The X Window System (also called X or X11) is the most common graphical display system for UNIX and Linux platforms. The user runs an X Server that renders the graphical applications, or X Clients, and interprets keyboard and mouse events and other input and output events. Each X Client is configured to connect most commonly using TCP/IP either on a local host, across an IP network or even across the Internet.

The X11 capture feature provides two areas of functionality:

- Encrypts X Windows communications to enhance security
- Provides a full session capture of every graphical session so the session can be logged and audited

To automatically allow X Clients to run, two items must be in place to locate the X Server and then authenticate to the service:

- The environment variable **DISPLAY**
- Authentication key

These items are commonly configured by the X Server into the users environment when they log in to the host. After you verify the items are in place, you can start using X11 forwarding.

DISPLAY Variable

The environment variable **DISPLAY** provides:

- The X Server location in the form of an IP address, host name, or fully qualified DNS domain name
- A display number that indicates the TCP/IP port that the X Server is listening on

Example

<code>DISPLAY=192.168.1.1:0.0</code>	Example of IPv4 and display 0
<code>DISPLAY=myhost:1.0</code>	Host name and display 1
<code>DISPLAY=myhost.mydomain.com:0.0</code>	Fully qualified address and display 0
<code>DISPLAY=[::1]:0.0</code>	IPv6 display 0

Each display on the X Server has its own display number, zero denotes TCP/IP port 6000, one is 6001, etc. This allows X Servers that have multiple displays to address each individually.

Authentication Key

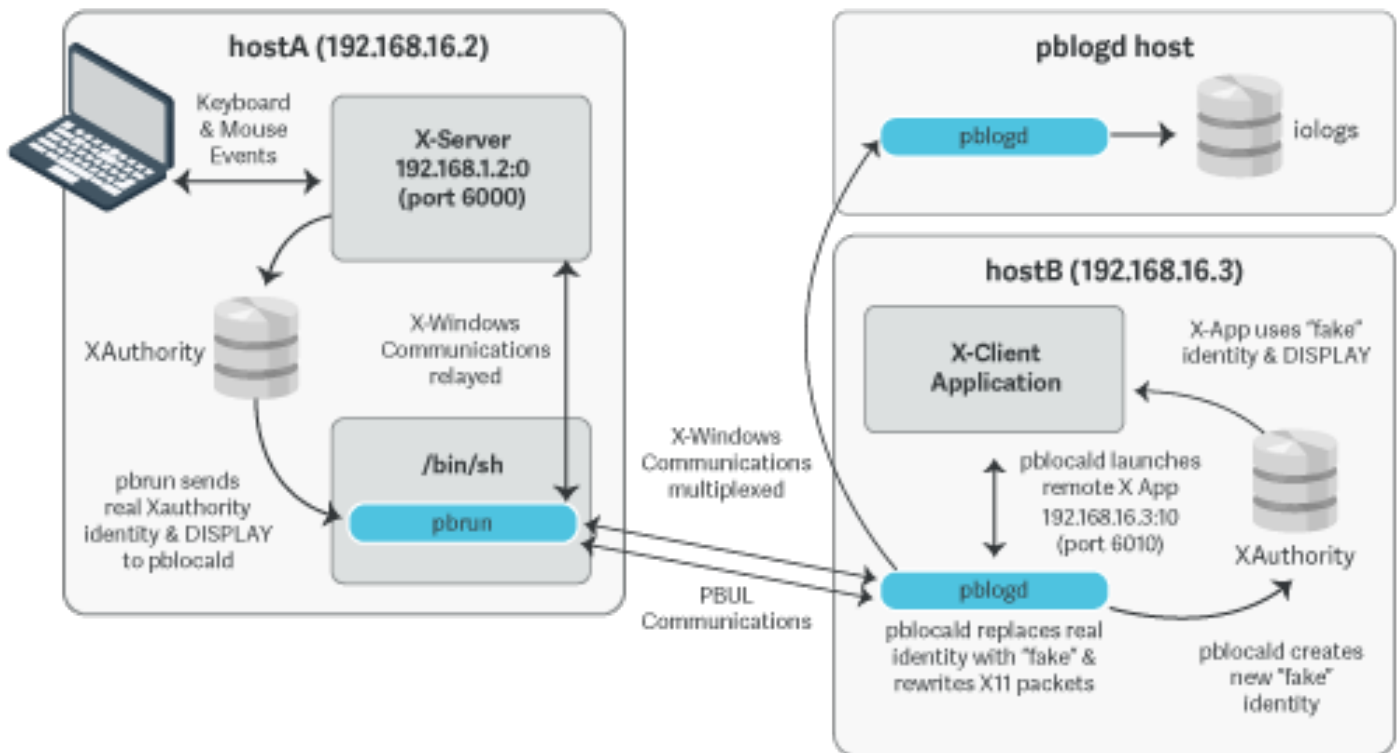
This preshared key authenticates each application. By default, a list of the keys is located in **\$HOME/.Xauthority**, and can be viewed and modified using the **xauth** command.


Example:

```
$ xauth list
pbuid/unix:16MIT-MAGIC-COOKIE-1 2c7ca46175aec0e4539b9e2119acff0d
```

The key is generally a randomly generated number created when the X Server session is created, and then used throughout the session by the X Clients.

Architecture Overview



X11 capture policy settings

When you run **pbrun -X** from the command line, by default, you can use the X11 forwarding, and the remote process is configured automatically to allow the X11 Client applications to run.

- **xwinforward (true or false):** Allow or deny X11 forwarding.
- **xwinreconnect (true or false):** The X11 tunnel is created between **pbrun** and **pblocald** to allow effective and efficient communications between the two ends of the tunnel. However, if there are networking considerations this option specifies that the tunnel is created using the existing Privilege Management for Unix and Linux network optimizing policy settings (for example, **noreconnect**).

If X11 forwarding is enabled it is only a matter of setting an iolog and the full X11 session is captured.

Viewing Session Capture Output

You can run **pbreplay** to output the captured X11 events. Events such as creating and destroying windows, textual window updates, text input, and mouse clicks are displayed as a summary alongside any output from the parent process.

In Privilege Management for Unix and Linux V8.5, **pbreplay** has a **-X** option added that you can use with the **-a** option to output the captured X11 events.



Example:

```
pbreplay -o -aX <path/to/iolog>
```



Note: You can use the following command to output all captured data. However, a large amount of data is displayed.

```
pbreplay -o -aXV <path/to/iolog>
```



Example: Summarized X11 output

```
pbreplay -o -aX /tmp/iolog.8TO9HF
[root@pbuild src]# gnome-terminal
{"request":"Session","cookie":"3dab3c6c01deb7c2e1109a96d830570b","proto":"MIT-MAGIC-COOKIE-1"}
{"reply":"Session","maj_vers":11,"min_vers":0}
{"request":"CreateWindow","chan":1,"window-id":98566145,"parent":612,"x":10,"y":10,"width":10,"height":10}
{"request":"ChangeProperty","chan":1,"window-id":98566145,"property":"WM_NAME","type":"STRING","value":"gnometerminal"}
{"request":"ChangeProperty","chan":1,"window-id":98566145,"property":"WM_ICON_NAME","type":"STRING","value":"gnome-terminal"}
{"request":"CreateWindow","chan":1,"window-id":98566146,"parent":98566145,"x":65535,"y":65535,"width":1,"height":1}
{"request":"ChangeProperty","chan":1,"window-id":98566145,"property":"WM_CLIENT_MACHINE","type":"STRING","value":"pbuild"}
{"request":"ChangeProperty","chan":1,"window-id":98566145,"property":"WM_COMMAND","type":"STRING","value":"gnometerminal"}
{"request":"ChangeProperty","chan":1,"window-id":98566145,"property":"WM_CLIENT_MACHINE","type":"STRING","value":"pbuild"}
{"request":"CreateWindow","chan":1,"window-id":98566147,"parent":612,"x":65436,"y":65436,"width":1,"height":1}
{"request":"ChangeProperty","chan":1,"window-id":98566147,"property":"WM_NAME","type":"STRING","value":"Fake Window"}
{"request":"Session","cookie":"3dab3c6c01deb7c2e1109a96d830570b","proto":"MIT-MAGIC-COOKIE-1"}
{"reply":"Session","maj_vers":11,"min_vers":0}
```



```
{ "request": "CreateWindow", "chan": 2, "window-
id": 102760449, "parent": 612, "x": 65516, "y": 65516, "width": 10, "height": 10}
{ "request": "Session", "cookie": "3dab3c6c01deb7c2e1109a96d830570b", "proto": "MIT-MAGIC-
COOKIE-1"}
{ "reply": "Session", "maj_vers": 11, "min_vers": 0}
{ "request": "DestroyWindow", "chan": 1, "window-id": 98566147}
{ "request": "CreateWindow", "chan": 1, "window-
id": 98566148, "parent": 612, "x": 0, "y": 0, "width": 658, "height": 438}
{ "request": "ChangeProperty", "chan": 1, "window-id": 98566148, "property": "WM_
NAME", "type": "STRING", "value": "Terminal"}
{ "request": "ChangeProperty", "chan": 1, "window-id": 98566148, "property": "WM_ICON_
NAME", "type": "STRING", "value": "Terminal"}
{ "request": "CreateWindow", "chan": 1, "window-
id": 98566149, "parent": 98566148, "x": 65535, "y": 65535, "width": 1, "height": 1}
{ "request": "ChangeProperty", "chan": 1, "window-id": 98566145, "property": "WM_
NAME", "type": "STRING", "value": "Terminal"}
{ "request": "ChangeProperty", "chan": 1, "window-id": 98566148, "property": "WM_CLIENT_
MACHINE", "type": "STRING", "value": "pbuild"}
{ "request": "ChangeProperty", "chan": 1, "window-id": 98566148, "property": "WM_ICON_
NAME", "type": "STRING", "value": "ctaylor@pbuild:~/pb/850a/pb/src"}
{ "request": "ChangeProperty", "chan": 1, "window-id": 98566148, "property": "WM_
NAME", "type": "STRING", "value": "ctaylor@pbuild:~/pb/850a/pb/src"}
{ "event": "ButtonPress", "chan": 1, "button": 1}
{ "event": "KeyPress", "chan": 1, "keysym": "l"}
{ "event": "KeyPress", "chan": 1, "keysym": "s"}
{ "event": "KeyPress", "chan": 1, "keysym": "Return"}
{ "request": "ChangeProperty", "chan": 1, "window-id": 98566148, "property": "WM_ICON_
NAME", "type": "STRING", "value": "ctaylor@pbuild:~/pb/850a/pb/src"}
{ "event": "KeyPress", "chan": 1, "keysym": "e"}
{ "event": "KeyPress", "chan": 1, "keysym": "x"}
{ "event": "KeyPress", "chan": 1, "keysym": "i"}
{ "event": "KeyPress", "chan": 1, "keysym": "t"}
{ "event": "KeyPress", "chan": 1, "keysym": "Return"}
{ "request": "DestroyWindow", "chan": 1, "window-id": 98566148}
```