



BeyondTrust

Privilege Management for Unix and Linux Sudo Manager 22.3 Administration Guide

Table of Contents

| | |
|---|-----------|
| Privilege Management for Unix and Linux Sudo Manager Introduction | 3 |
| Overview | 4 |
| Sudo Manager Component, Directory, and File Locations | 4 |
| Sudo Manager Policy Server | 5 |
| Install Sudo Manager Policy Server | 5 |
| Configure Sudo Manager Policy Server | 6 |
| Sudo Manager Plugin | 8 |
| Install Sudo Manager Plugin | 8 |
| Configure Sudo Manager Plugin | 9 |
| Log Server for Sudo Manager | 11 |
| Central Management of Sudoers Policies on Sudo Manager Policy Server | 12 |
| Host Aliases | 13 |
| REST API for Sudo Manager | 15 |
| Sudo Manager Client Settings | 16 |
| Additional Information | 36 |

Privilege Management for Unix and Linux Sudo Manager

Introduction

Sudo is widely used by many organizations to define and delegate elevated privileges throughout their Unix and Linux systems. Its appeal lies in the additional layer of protection it gives to root access while providing logging and auditing features, all with no upfront cost.

However, sudo's limitations become apparent when deployed in larger environments because it does not scale well within an enterprise. It does not provide central storage and administration of sudoers policy files. It does not provide a secure and efficient means of distributing sudoers policy files over multiple systems. It does not natively protect the integrity of generated logs and cannot provide remote logging to remote servers, which are best practices for security and compliance.

Sudo alternatives, such as Privilege Management for Unix and Linux, are commercially available to provide a more complete, seamless, and secure least privilege solution for the enterprise that addresses the aforementioned issues and more. This upgrade, however, entails an investment of time and resources.

For organizations that choose not to fully convert their sudo-managed systems, BeyondTrust offers Privilege Management for Unix and Linux Sudo Manager, hereinafter Sudo Manager, which simplifies and enhances sudo management using some of the core features of Privilege Management for Unix and Linux. This allows for a quick and cost-effective implementation and continued use of all existing sudoers files.

Sudo Manager is BeyondTrust's offering to provide better management and maintenance of sudo's files and data, leveraging some of the rich core features of Privilege Management for Unix and Linux without replacing sudo itself. Implementing Sudo Manager has the following benefits:

- **Centralization of sudoers policies:** Policies are stored in a secure database on the Policy Server host.
- **Change management for sudoers policies:** Once sudo policies are stored on the Policy Server, they can be checked out, modified, and checked back in centrally, without the need to go to each sudo host.
- **Integration with Privilege Management for Unix and Linux event logs:** After policy processing, an accept or reject event is logged in the event log.



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.



IMPORTANT!

The BeyondInsight integration for Privilege Management for Unix and Linux is no longer supported. Instead, PMUL uses BeyondInsight for Unix & Linux and ElasticSearch.

**IMPORTANT!**

Both **pbsguid** and **pbsguid** are deprecated as of PMUL version 22.3.0.

Overview

To effectively administer Sudo Manager, it is necessary to understand how the product works. A typical Sudo Manager configuration consists of the following:

- **pbsudomgr.so**: The plugin extending sudo with some of the core features of Privilege Management for Unix and Linux
- Sudo Manager **Policy Server**: The component providing central management of sudoers files
- **Log Host**: The component writing the event logs
- **pbadmin**: A robust command line utility for administrators to manage files and data used by Privilege Management for Unix and Linux Sudo Manager

The **pbsudomgr.so** plugin must reside on the sudo hosts being managed. For optimal security, the Sudo Manager Policy Server and log host should be separate machines isolated from normal activity.

Sudo Manager 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 Manager Policy Server

Sudo Manager Policy Server is the central repository of the sudoers policy files. We highly recommend that hosts designated as Sudo Manager Policy Servers be isolated from regular user activity to shield policies from users that can elevate their privileges.

Whenever Sudo Manager is installed on a sudo client host, a copy of the sudoers file, and any included policy files, are sent via encrypted file transfer to Sudo Manager Policy Server where they are imported into a SQLite database. Subsequently, whenever sudo runs on a sudo client host, it ensures that it has the latest copy of the file(s) from Sudo Manager Policy Server. This centralization of the sudoers files gives you better control over the integrity and consistency of the policies to be used across your organization. Modification of policy files is made against a singular location, with tools to check out a file from the Policy Server's database and to check it back in when edits are done. The policy changes are automatically distributed to appropriate hosts when the file gets pulled down at each sudo invocation at the target host, or by on-demand request.

Install Sudo Manager Policy Server

Sudo Manager Policy Server is installed using the **pbinstall** program.

When you run **pbinstall**, answer **yes** to the install menu:

Install Sudo Policy Server?



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`.



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>.

Create an Appid and Appkey

The installation program for the Sudo Manager Policy Server creates an application ID (**appid**) and application key (**appkey**), which are used during the client registration of Sudo Manager hosts. The **appid** and **appkey** can be manually created:

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

Create a Registration Profile

When installing the Sudo Manager Policy Server, a default profile **sudodefault** is created by **pbinstall** and the file `/etc/pbsudo.settings.default` is generated. When installing Sudo Manager on sudo hosts, this **sudodefault** profile, in conjunction with the aforementioned **appid** and **appkey**, can be used during the required client registration portion of the installation. However, you can also create your own registration profile. First, create the `/etc/pbsudo.settings.<name>` (where **name** is a name to identify this specific sudo settings file). This file will be used in your registration profile and should contain the following settings that you need to copy from `/etc/pb.settings`:

- **restkeyencryption**
- **pbrestport**

- **submitmasters**
- **transparentfailover**
- **logport**
- **logservers**
- **logserverdelay**
- **logserverprotocoltimeout**
- **randomizelogservers**
- **minoutgoingport**
- **maxoutgoingport**
- **networkencryption**
- **enforcehighsecurity**
- **ssl**
- **ssloptions**
- **sslengine**
- **sslcountrycode**
- **sslprovince**
- **ssllocality**
- **sslorgunit**
- **sslorganization**
- **registrynameservice**
- **sslengine**
- **sslpbruncipherlist**
- **pbsudofailovertimeout**
- **pbsudorefresh**
- **pbsudofailover**

Create the registration profile by running the following command on the Sudo Manager Policy Server as **root**:

```
# pbdbutil --reg -u '{"name":"<profile_name>","data":
[{"type":"save","to":"/etc/pbsudo.settings","fname":"/etc/pbsudo.settings.<name>"},
{"type":"save","sname":"networkencryption"},
{"type":"save","sname":"restkeyencryption"},
{"type":"save","sname":"sslservercertfile"}]}'
```

Add the **pbsudo.settings.<name>** to the configuration database by running:

```
# pbdbutil --cfg -l /etc/pbsudo.settings.<name>
```

Configure Sudo Manager Policy Server

After the installation, the configuration file **/etc/pb.settings** is created for Sudo Manager Policy Server. The file **/etc/pbsudo.settings.default** is also created, to be used when registering a Sudo Manager client host with this Policy Server.

The following settings keywords are added to the **/etc/pb.settings**:

sudoersdb

The filename and location of the SQLite database where the sudoers files are stored.



Example:

```
sudoersdb /mypath/pbsudo.db
```

Default

```
sudoersdb /opt/<prefix>pbul<suffix>/dbs/pbsudo.db
```

sudoersdir

The absolute path of the directory which Sudo Manager Policy Server will use to export and import sudoers file. Sudoers and included files can be checked out, edited, and checked in using the existing mechanism in **pbdbutil**, within the **--sudo** option.



Example:

```
sudoersdir /mypath/sudoersdir
```

Default

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

Sudo Manager Plugin

Sudo Manager provides a plugin that fits into sudo's modular framework to handle security policy processing. This plugin, along with supporting components and utilities, makes up Sudo Manager and must be installed on all sudo hosts whose files you want to manage (sudoers files and generated data). During installation, the active sudoers files from this host are securely transported to the centralized Sudo Manager Policy Server repository for storage and maintenance, and the local copy is effectively disabled.



IMPORTANT!

Once the sudo policy is in the Sudo Manager Policy Server host and the Sudo Manager policy plugin is specified in **sudo.conf**, any edits to the sudo client's **/etc/sudoers** or its included files are ignored. The changes to sudoers policies are implemented on the Sudo Manager Policy Server by checking out the sudoers files, making the changes, and checking them back in.

Whenever sudo is invoked on the target host to run a command, Sudo Manager on that host first ensures that it has the latest sudoers file from the Sudo Manager Policy Server before it proceeds into the policy processing. It saves a cached copy of the sudoers policy file so that users are never prevented from issuing sudo commands in case network issues arise. The cached policy remains valid for a configurable amount of time before the next update from the Policy Server is warranted at the next sudo invocation.



IMPORTANT!

Use the keyword **pbsudofailover** to enable and disable using the cached policy. By default, this keyword is set to **no**. If you want to allow the Sudo Manager client to fail over to the cached policy when connection to all Sudo Manager policy servers, or **logservers** fails, set **pbsudofailover** to **yes** in **/etc/pbsudo.settings**.

You can set it to **yes** in **pbsudo.settings.default**, so any new Sudo Manager client installation will have it set to **yes** in its **pbsudo.settings** file.



Note: Sudo password validation occurs after policy processing, so any password errors are not recorded as "rejects" in the Sudo Manager event log. A finish event is generated with the **exitstatus** "ConfirmUser <username> failed".



For more information, please see the following:

- "[Central Management of Sudoers Policies on Sudo Manager Policy Server](#)" on page 12
- "[pbsudorefresh](#)" on page 34

Install Sudo Manager Plugin

Sudo Manager must be installed on all sudo hosts. The minimum required version of sudo is v1.8.23, but the policy constructs in the sudoers file must be available in v1.9.0 or earlier. The installation requires client registration. You will also need Sudo Manager Policy Server's Application ID, Application Key, Client Profile name, as well as the hostname and port for a REST service.

The **sudomgrinstall** installer program registers the target sudo host and securely transfers the sudoers policy file, along with relevant include files, to the Sudo Manager Policy Server for storage and maintenance. It then lays down Sudo Manager's customized policy plugin (**pbsudomgr.so**), hooking it into the sudo front end configuration (**sudo.conf**), simultaneously deactivating any preexisting plugins for policy processing.

i For more information, please see the following:

- [Privilege Management for Unix and Linux Sudo Manager 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>
- "Host Aliases" on page 13
- "Central Management of Sudoers Policies on Sudo Manager Policy Server" on page 12

Configure Sudo Manager Plugin

After installation, the configuration file **/etc/pbsudo.settings** is created with the necessary information that Sudo Manager needs to function properly: (for example, identify the logservers).

The following keywords described below are important settings for Sudo Manager.

Sudo Manager Required Settings

- submitmasters
- enforcehighsecurity
- logport
- logservers
- networkencryption
- pbrestport
- restkeyencryption
- sharedlibcurldependencies
- sharedlibssldependencies

Optional Sudo Manager Settings

- pbsudofailovertimeout
- pbsudorefresh
- registrynameservice
- sslengine
- sslpbruncipherlist
- pbsudofailover



For more information, please see the following:

- *"enforcehighsecurity" on page 24*
- *"logport" on page 18*
- *"logservers" on page 18*
- *"networkencryption" on page 22*
- *"pbrestport" on page 16*
- *"restkeyencryption" on page 16*
- *"pbsudofailovertimeout" on page 33*
- *"pbsudorefresh" on page 34*

Log Server for Sudo Manager

Sudo Manager employs one or more log servers to centrally store and manage audit data. Establishing dedicated central log servers with data encryption capabilities and keeping them separate from hosts where privileged users run daily tasks assures the integrity of your audit trail. Sudo Manager hosts identify their log servers by the **logservers** keyword in the product settings file **/etc/pbsudo.settings**.

After Sudo Manager plugin processes a sudo command on the target host and produces either an accept or reject, it securely transmits the event log records directly to the log server for writing.

The event log files may be encrypted for added security.

When a Sudo Manager log server cannot be reached, the sudo event logging mechanism is used.

With the event logs stored on the Sudo Manager log server, you can take advantage of the rich features that comes with the product.

- Integrate with Splunk
- Forward accept/reject events to BeyondInsight, if integrated, and have access to additional log reporting and analysis tools.
- Integration with ElasticSearch/LogStash

Install Log Server for Sudo Manager

The log server is installed using the **pbinstall** program.

When you run **pbinstall**, answer **yes** to the install menu:

```
Install Log Host?
```



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>.

Central Management of Sudoers Policies on Sudo Manager Policy Server

When sudoers policies are uploaded to a Sudo Manager Policy Server, they are stored in a SQLite database in the file and directory specified by `sudoersdb` settings, and the local sudoers policies on the sudo hosts with Sudo Manager Policy Server are no longer used when running `sudo` commands on these hosts.

Sudoers policies can be managed from any client or server within the Sudo Manager enterprise. The command line utility can be used directly on a Sudo Manager Policy Server, or the user can specify the `--client` argument to the `pbdbutil --sudo` calls to remotely administer the sudo policies.

To manage the sudoers policies, export the file, modify it, and re-import it using `pbdbutil --sudo -e` and `pbdbutil --sudo -i` commands.

Export Specified Sudoer Policy File from Database (`pbdbutil --sudo -e`)



Note: For export, if you use `--force` option, the target directory structure is created. If you do not want to use force, you can manually create the directories before export.

```
# pbdbutil --sudo -e sudohost.bt.com@/etc/sudoers --force
```

If there are multiple versions of a file in the database, an export with `-V` option checks out the file with specified version:

```
# pbdbutil --sudo -l -l /etc/sudoers
{"pathname": "sudohost.bt.com@/etc/sudoers", "version": 1, "tag": "nu ll", "deleted": 0, "created": "2015-07-07 12:15:47"}
{"pathname": "sudohost.bt.com@/etc/sudoers", "version": 2, "tag": "nu ll", "deleted": 0, "created": "2015-07-07 12:26:59"}
# pbdbutil --sudo -e -V "2" sudohost.bt.com@/etc/sudoers --force
```



Example: Usage of Import (`pbdbutil --sudo -i`):

```
# pbdbutil --sudo -i /etc/pbsudoers/sudohost.bt.com/etc/sudoers
```



For more examples, please see **Sudo Database Options** in *Privilege Management for Unix and Linux Administration Guide* at <https://www.beyondtrust.com/docs/privilege-management/unix-linux/admin/index.htm>.

Host Aliases

A Sudo Manager host alias provides a way to group several hosts that share a common set of sudoers policies. This is not the same as a Sudoers Host_Alias, which is resolved inside a sudoers policy. Rather, a Sudo Manager host alias is used by the Sudo Manager Policy Server to select an appropriate policy by mapping different hostnames to the same arbitrary string.

Host aliases are manipulated using the **pbdbutil** command on the sudo Policy Server:

- To map host names to an alias:

```
pbdbutil --sudo -A <ALIAS> <hostnames...>
```

- To unmap host names from an alias:

```
pbdbutil --sudo -X <ALIAS> <hostname...>
```

- To list current aliases:

```
pbdbutil --sudo -G [pattern]
```

Sudo policies are stored in a database table that is keyed on path name and version. The version increments whenever a new copy is uploaded. The path name is composed of two parts:

```
<fqdn>@/<sudoers_path>
```

- The client's fully qualified domain name
- The client's default sudoers path (usually **/etc/sudoers**) as reported by **sudo -V**



Example: If your sudo Policy Server defines **sudoersdir** in **/etc/pb.settings** as **/etc/pbsudoers**, and your Linux client's name is **client0.bt.net**, then the pathname to identify its policy is:

```
client0.bt.net@/etc/sudoers
```

By default, each new client's policy is uploaded to the sudo Policy Server to a unique path during installation after client registration.



Note: The default sudoers path can vary by host. While it is commonly **/etc/sudoers**, third-party versions and custom builds may use different paths, like **/usr/local/etc/sudoers**. Use **sudo -V** to discover the current path. Each alias should be used only for hosts with the same sudoers path. Hosts with the same sudoers policy but a different sudoers path should use a different alias.



Note: The client governs the right part following the hostname or alias. The **pbsudomgr.so** plugin defaults to **/etc/sudoers** on all platforms. Use the **sudoers_file** plugin argument to specify a different path.

A Privilege Management for Unix and Linux Sudo Manager host alias can map several hosts to the same policy by substituting an arbitrary ALIAS for the hostname:

```
<ALIAS>@/<sudoers_path>
```



Note: *pbsudoinstall* can easily create host aliases during installation on sudo hosts. For more information, please see the [Privilege Management for Unix and Linux Sudo Manager 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>.

To implement an alias with policies outside of **pbsudoinstall**, first create a policy for the alias, then assign hosts to the alias. Do all of this on the sudo Policy Server. This can be done before or after the clients are installed, and reinstallation of the clients is not required.

1. Determine the **sudoersdir** path defined in the sudo Policy Server's settings:

```
# pbcall -getstringsetting sudoersdir
```

We use **/etc/pbsudoers** in the following steps.

2. Invent an alias name in all uppercase letters. For simplicity, we use **ALIAS**.
3. Determine the default sudoers file path. For familiarity, we use **/etc/sudoers**.
4. Create the policy directory and sudoers file.

```
# mkdir -p /etc/pbsudoers/ALIAS/etc
# vi /etc/pbsudoers/ALIAS/etc/sudoers
```

5. Verify the policy syntax with **visudo**, if available.

```
# visudo -c -f /etc/pbsudoers/ALIAS/etc/sudoers
```

6. Import the policy into the PBDB.

```
# pbdbutil --sudo -i /etc/pbsudoers/ALIAS/etc/sudoers
# pbdbutil --sudo -l
```

7. Map clients to the alias.

```
# pbdbutil --sudo -A ALIAS <host1> <host2> ...
# pbdbutil --sudo -G ALIAS
```

8. Adjust the **sudoers_file** plugin argument in **sudo.conf** on all clients, especially those platforms with third-party or custom sudo. For example:

```
# grep Plugin /etc/sudo.conf
Plugin sudoers_policy /usr/lib/beyondtrust/pb/pbsudoers.so sudoers_file=/etc/sudoers
Plugin sudoers_io /usr/lib/beyondtrust/pb/pbsudoers.so
```

9. Verify the configuration on each client:

```
# sudo -V; sudo -l
```

REST API for Sudo Manager

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, and policy configuration 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 in the *Privilege Management for Unix and Linux Administration Guide* at <https://www.beyondtrust.com/docs/privilege-management/unix-linux/admin/index.htm>.

Sudo Manager Client Settings

This section provides settings and configuration options specific to Sudo Manager plugin client.



For a comprehensive listing of settings available, please see [Settings](#) in the *Privilege Management for Unix and Linux Administration Guide*.

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

- Sudo Manager client



For more information, please see "[networkencryption](#)" on page 22.

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

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.

This keyword is ignored if **pbsudofailover** is set to **no**. When **pbsudofailover** is set to **no**, sudo fails if it cannot connect to policy or logserver, and displays an error regardless of the value of **transparentfailover**.

**Example:**

```
transparentfailover yes
```

Default

```
transparentfailover yes
```

Used on

Sudo Manager client

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).



Example:

```
logport 12345
logport pblogd
```

Default

```
logport 24347
```

Used on

- Sudo Manager policy server
- Sudo Manager client

logservers

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

The **logservers** setting provides a list of outgoing connection information for Sudo Manager 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

- Sudo Manager policy server
- Sudo Manager client

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

- Sudo Manager policy server
- Sudo Manager client

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

- Sudo Manager policy server
- Sudo Manager client

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

- Sudo Manager policy server
- Sudo Manager client

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

- Sudo Manager policy server
- Sudo Manager client

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

- Sudo Manager policy server
- Sudo Manager client

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

- Sudo Manager policy server
- Sudo Manager client



For more information, please see the following:

- ["ssl" on page 25](#)
- ["ssloptions" on page 26](#)
- ["sslcountrycode" on page 29](#)
- ["sslprovince" on page 29](#)
- ["ssllocality" on page 30](#)
- ["sslorgunit" on page 31](#)
- ["sslorganization" on page 31](#)

ssl

When set to **yes**, the **ssl** setting enables the use of Privilege Management for Unix and Linux SSL features.

**Example:**

```
ssl yes
```



Note: As of PMUL 22.3, **ssl no** is deprecated. PMUL will always use SSL.

Default

```
ssl yes
```

Used on

- Sudo Manager policy server
- Sudo Manager client


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 ssloptions line. |
| AllowNonSSL | To communicate with older, non-SSL versions of Privilege Management for Unix and Linux, add AllowNonSSL to your ssloptions 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. |
| TLSTMinV1.0, TLSTMinV1.1, TLSTMinV1.2, TLSTMinV1.3 | When SSL is enabled, this option allows you to set the minimum SSL/TLS value to use in the protocol. |
| TLSTMaxV1.0, TLSTMaxV1.1, TLSTMaxV1.2, TLSTMaxV1.3 | When SSL is enabled, this option allows you to set the maximum SSL/TLS value to use in the protocol. |
| RequireSSL | 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. 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. |
| SSLFirst | If the SSLFirst option is selected, this option forces the SSL handshake to happen before the Privilege Management for Unix and Linux handshake. The SSLFirst option must be set on every Privilege Management for Unix and Linux host including clients and servers. The SSLFirst option is turned on by default in version 10.3.2 and later. |
| sslverbose | If the sslverbose option is selected, server components log informational messages that are sent to |

| Option | Description |
|-----------------------|---|
| | <p>error logs, detailing connections, SSL/TLS protocols, and the encryption ciphers used to communicate. This is a debugging and diagnostic option</p> |
| 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 sslservercertfile and sslserverkeyfile keywords, specifying the server's certificate and key. This feature alternatively uses the sslservercertdir, sslserverkeydir, and sslpbbruncadir 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

- Sudo Manager policy server
- Sudo Manager client

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

- Sudo Manager policy server

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 25](#)
- ["sslprovince" on page 29](#)
- ["ssllocality" on page 30](#)
- ["sslorgunit" on page 31](#)
- ["sslorganization" on page 31](#)

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 25](#)
- ["sslcountrycode" on page 29](#)
- ["ssllocality" on page 30](#)
- ["sslorgunit" on page 31](#)
- ["sslorganization" on page 31](#)

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

i For more information, please see the following:

- ["ssl" on page 25](#)
- ["sslcountrycode" on page 29](#)
- ["sslprovince" on page 29](#)
- ["sslorgunit" on page 31](#)
- ["sslorganization" on page 31](#)

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

i For more information, please see the following:

- ["ssl" on page 25](#)
- ["sslcountrycode" on page 29](#)
- ["sslprovince" on page 29](#)
- ["ssllocality" on page 30](#)
- ["sslorganization" on page 31](#)

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 25](#)
- ["sslcountrycode" on page 29](#)
- ["sslprovince" on page 29](#)
- ["ssllocality" on page 30](#)
- ["sslorgunit" on page 31](#)

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

sslprncipherlist

OpenSSL provides a variety of algorithms that can be used for encryption. The **sslprncipherlist** 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 **sslprncipherlist** 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 **sslprncipherlist** keyword, the order of cipher lists is not relevant.

This format: **sslprncipherlist cipherlist= TLSv1.2:!SSLv2:@STRENGTH tlsv1.3= TLS_AES_256_GCM_SHA384**

is the same as this format:

sslprncipherlist 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.



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 **sslprncipherlist**. To use more than one cipher set, separate the values with colons.

pbsudofailovertimeout

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

When multiple Sudo Manager policy servers are configured in **/etc/pbsudo.settings** on sudo clients, they automatically try to retrieve their policy from each server in turn, providing failover. The **pbsudofailovertimeout** setting specifies how long it will take to timeout and try the next host in the list.



Example:

```
pbsudofailovertimeout    60
```

Default

```
pbsudofailovertimeout    30
```

Used On

Sudo Manager client hosts

pbsudorefresh

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

Sudo Manager maintains a cached copy of the **sudoers** file on the target host which it updates periodically. The **pbsudorefresh** setting is the time in seconds that a client's sudoers cache is valid. After that time, it contacts the policy server to refresh the cache.



Example:

```
pbsudorefresh 120
```

Default

```
pbsudorefresh 30
```

Used On

pbsudo client (stored in profile on policy server)

pbsudofailover

- **Version 22.2.0 and earlier:** **pbsudofailover** setting not available.
- **Version 22.2.0 and later:** **pbsudofailover** setting available.

When set to **yes**, allows sudo to run the secured task when the policy or log server cannot be contacted. The **transparentfailover** keyword is ignored so that connection errors are always displayed.

When set to **no**, an error message displays indicating sudo cannot contact the policy server or log server regardless of the value on **transparentfailover**.



Example:

```
pbsudofailover yes
```



IMPORTANT!

Use the keyword **pbsudofailover** to enable and disable using the cached policy. By default, this keyword is set to **no**. If you want to allow the Sudo Manager client to fail over to the cached policy when connection to all Sudo Manager policy servers, or **logservers** fails, set **pbsudofailover** to **yes** in **/etc/pbsudo.settings**.

You can set it to **yes** in `pbsudo.settings.default`, so any new Sudo Manager client installation will have it set to **yes** in its `pbsudo.settings` file.

Default

```
pbsudofailover no
```

Used On

Sudo Manager client hosts

Additional Information

The *Privilege Management for Unix and Linux Administration Guide* provides information about features not detailed in this guide. Refer to the links below for those topics.



For more information, please see:

- [Manage and Test System Configuration](#)
- [License Management](#)
- [Registry Name Service and Database Synchronization](#)
- [Logging](#)
- [Message Router Troubleshooting](#)
- [Firewalls](#)
- [System Upgrades](#)
- [Administration Programs](#)
- [REST API](#)