Safeguard for Sudo 7.2

Administration Guide

Legend

WARNING: A WARNING icon highlights a potential risk of bodily injury or property damage, for which industry-standard safety precautions are advised. This icon is often associated with electrical hazards related to hardware.

CAUTION: A CAUTION icon indicates potential damage to hardware or loss of data if instructions are not followed.

Safeguard for Sudo Administration Guide
Updated - 22 November 2021, 03:00
Version - 7.2

Contents

1

About this guide

Welcome to the *One Identity Safeguard for Sudo Administration Guide*. This guide is intended for Windows, Unix*, Linux, and Macintosh system administrators, network administrators, consultants, analysts, and any other IT professional who will be installing and configuring Safeguard for Sudo for the first time.

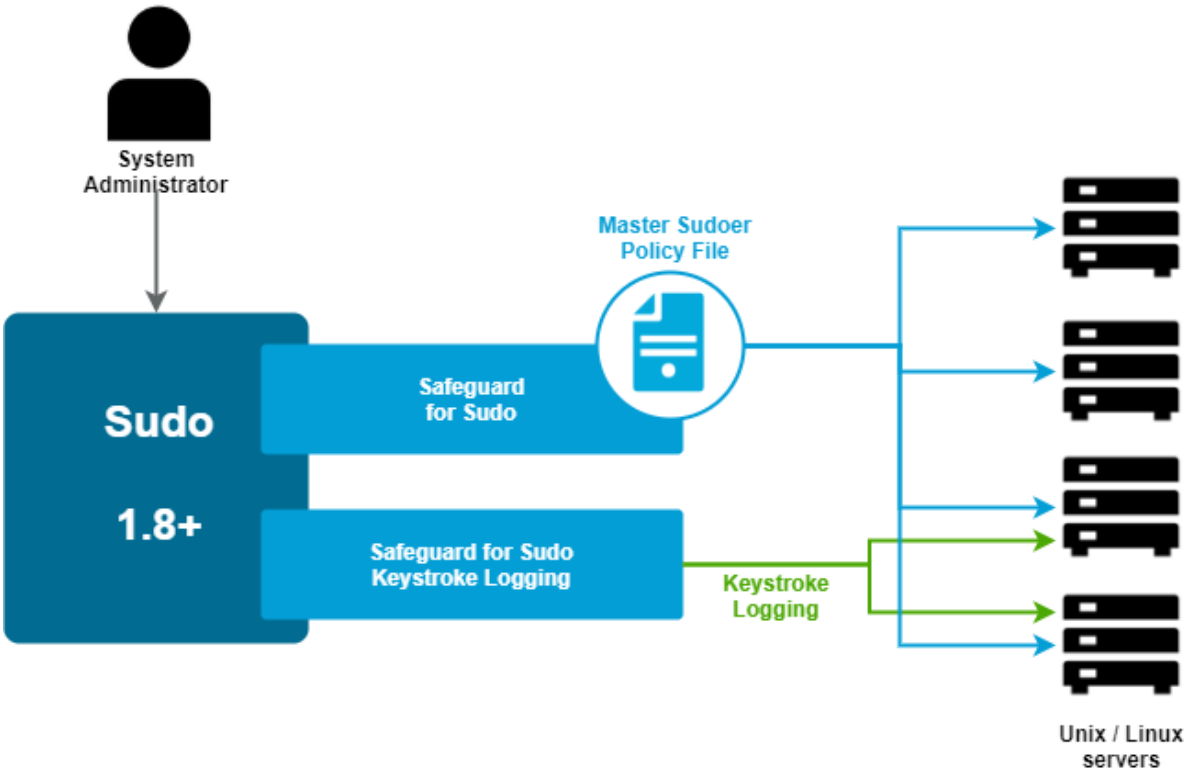
This guide explains how to install and configure Safeguard components for the sudo and pmpolicy policy types from the command line.

* The term "Unix" is used informally throughout the Safeguard documentation to denote any operating system that closely resembles the trademarked system, UNIX.

Introducing Safeguard for Sudo

Safeguard for Sudo helps Unix/Linux organizations take privileged account management through sudo to the next level: with a central policy server, centralized management of sudo and sudoers, centralized reporting on sudoers and elevated rights activities, event and keystroke logging of activities performed through sudo, and offline policy evaluation. With Safeguard for Sudo, One Identity provides a plugin to Sudo 1.8.1 (and later) to make administering sudo across a few, dozens, hundreds, or thousands of Unix/Linux servers easy, intuitive, and consistent. It eliminates the box-by-box management of sudo that is the source of so much inefficiency and inconsistency. In addition, the centralized approach delivers the ability to report on the change history of the sudoers policy file.

Figure 1: Safeguard for Sudo Architecture



Safeguard for Sudo enables you to get more value, security, and compliance out of your existing investment in sudo across any number of Unix/Linux systems.

Features and benefits of Safeguard for Sudo

Embracing and enhancing Sudo

The vast majority of organizations with Unix/Linux machines in their infrastructure use the open-source sudo project to help delegate the Unix *root* account to achieve privileged account management objectives. Sudo has a proven history of delivering value, however, management of sudo can be cumbersome, sudo policy across multiple servers is often inconsistently written and executed, and sudo does not include the ability to centrally manage the sudoers policy on multiple systems that is so critical to security and compliance initiatives. One Identity LLC, the company that pioneered the "Active Directory bridge" market with Authentication Services, continues to lead the way for identity and access management in Unix environments, with powerful and innovative new capabilities that provide enterprise-level privileged account management (PAM) by enhancing an existing sudo installation with centralized policy, reporting, management, and keystroke logging through Safeguard for Sudo.

Safeguard for Sudo provides powerful capabilities:

- Centralized management of sudo across any number of Unix/Linux servers
- Centralized reporting on sudo policy, activities, and history
- The ability to join a policy server in pmpolicy mode
- Event and keystroke logging
- Offline policy evaluation and log synchronization
- Policy revision management with change tracking and reporting, and policy roll-back
- Support for multiple sudoers policies for each server

Extend Sudo

Safeguard for Sudo enhances sudo with new capabilities (central policy server and keystroke logging) that embrace and extend sudo through the Sudo Plugin which fits into the Sudo modular architecture.

Central Sudo policy

Safeguard for Sudo permits sudo to use a central service to enforce a policy, removing the need for administrators to manage the deployment of the sudoers policy file on every system. This improves security and reduces administrative effort by centrally administering sudo policy for privileged account management across any number of Unix/Linux servers.

Safeguard for Sudo also offers the ability to join a policy server in pmpolicy mode. The pmpolicy mode supports a script-style policy format that can be used to build custom security policies with fine-grained control of privileges.

Event logging

The Safeguard for Sudo event logging feature provides the ability to log all commands performed through sudo to know which commands were accepted and rejected, who performed the command, and when the command was performed.

Keystroke logging

The Safeguard for Sudo keystroke logging feature provides the ability to log keystrokes, then view and replay keystroke logs for end-users that perform activities through sudo. The keystroke log provides a comprehensive view of what activities were performed and the commands that were run across all systems. You can filter the report in many ways to find data quickly. For example, you can filter on specific commands or for commands run during a specific time period.

Audit server logging

Administrators can stream event logs and keystroke (IO) logs from a client to a sudo log audit server (or compatible server). A syslog output of streamed keystroke (IO) logs can be used to send the data to a Security Information and Event Management (SIEM) tool.

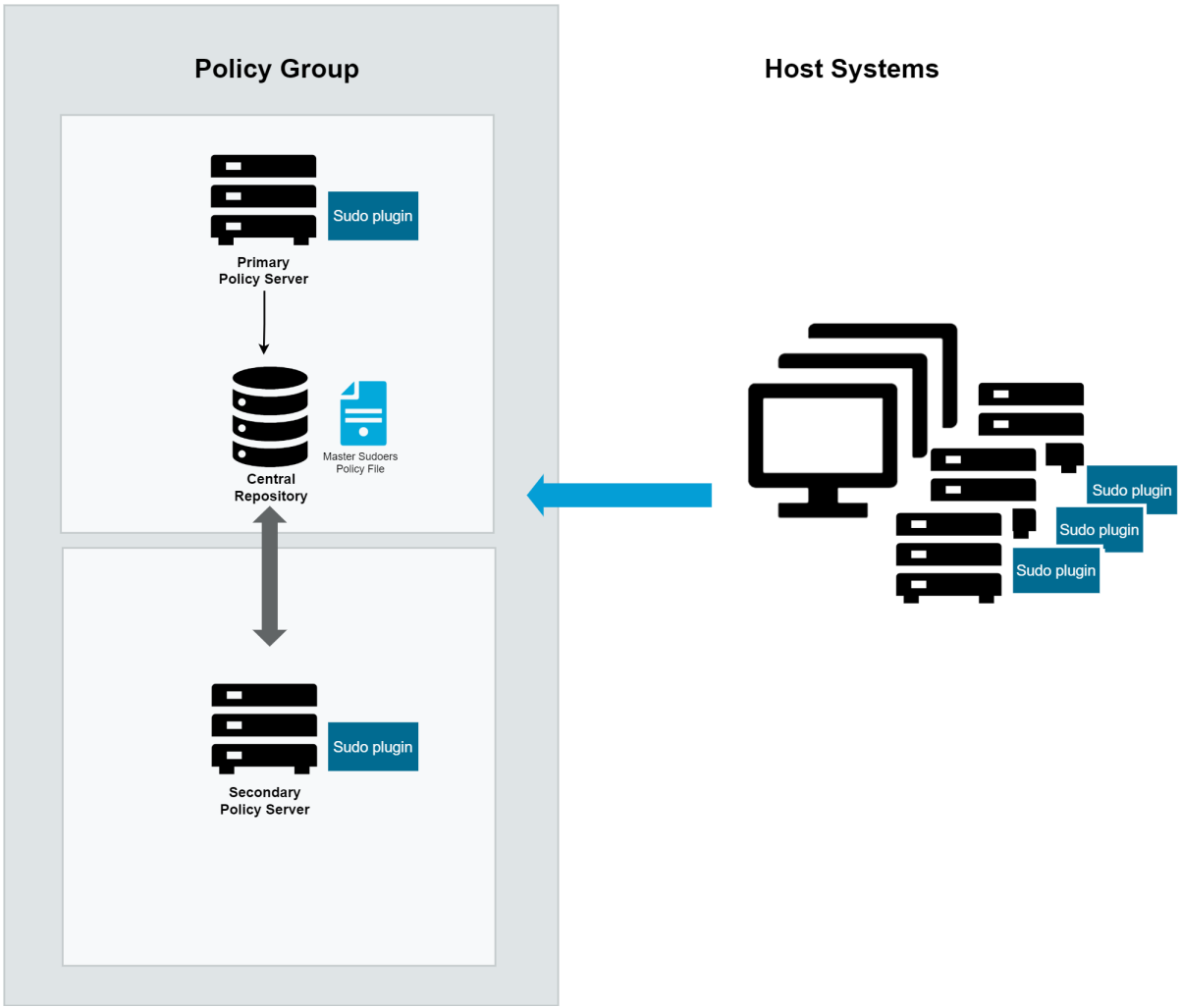
Offline policy evaluation and log synchronization

Safeguard for Sudo supports offline policy caching. When a Sudo Plugin host operates offline, it stores all log files on the host, then synchronizes the log data back to the primary policy server when it becomes available. See Safeguard for Sudo Policy Evaluation on page 197 for more information.

How Safeguard for Sudo works

A basic Safeguard for Sudo configuration would include a primary and a secondary policy server, (known as a policy group), and any number of hosts with the Sudo Plugin installed.

Figure 2: How Safeguard for Sudo Works



The first policy server configured is the primary policy server which holds the master copy of the sudoers policy. Additional policy servers configured in the policy group are secondary policy servers. The primary policy server and any number of additional secondary policy servers share the common sudoers policy.

The Sudo Plugin is installed on each host system. Then the hosts are joined to the policy group. Once joined, sudo commands that run on the hosts are sent to the primary policy server to be evaluated against the centralized policy. (**Note**: The local sudoers files (/etc/sudoers and /etc/sudoers.d) are no longer used to evaluate the sudo policy on joined hosts.)

The primary policy server either accepts or rejects the commands; that is, the primary policy server either allows the command to run on the host or not. The primary policy server records an event each time a command is accepted or rejected. And, if enabled for keystroke logging, the primary policy server records the keystrokes entered on the hosts.

3

Planning Deployment

Before you run the installer, consider the following questions:

1. Which machines in your network will run policy servers?

   If you only plan to use one policy server for an entire network, it should be the most reliable and secure machine.

   You can specify multiple policy servers to avoid having a single point of failure.

   If more than 150 users will be using a single `pmmasterd` for validation, you will want to have multiple policy servers to avoid a UNIX network resource bottleneck. Plan to have a maximum of 150 users validating at a single policy server.

2. Which machines will be managed hosts?

   Only those hosts running the Sudo Plugin may receive and run Safeguard for Sudo requests.

   One Identity recommends that you initially specify one policy server and three or four Sudo Plugin hosts when you first install and experiment with Safeguard for Sudo.

3. What level of protection do you require?

   If you require greater protection, you can select an encryption level such as AES, or a dedicated encryption system such as Kerberos. When configuring Safeguard in interactive mode, you are asked if you are using Kerberos. If you are using Kerberos, Safeguard automatically uses Kerberos for encryption.

4. What port number should `pmmasterd` use to listen for network requests?

   Choose numbers that do not conflict with other numbers in the `/etc/services` file. Ensure these entries are propagated to all machines accessing Safeguard.

5. Which directory should contain the Safeguard log files?

   By default, the log files are placed in `/var/adm` or `/var/log` depending on the host architecture. The installer allows you to change the directory by specifying command line options to the Safeguard daemons. The partition needs to contain enough space for log files to increase in size.

System requirements

Prior to installing Safeguard, ensure your system meets the minimum hardware and software requirements for your platform.

Table 1: Hardware and software requirements

Component	Requirements
Operating systems	See Supported platforms to review a list of platforms that support Safeguard clients.
Disk space	80 MB of disk space for program binaries and manuals for each architecture. Considerations: • At a minimum, you must have 80 MB of free disk space. The directories in which the binaries are installed must have sufficient disk space available on a local disk drive rather than a network drive. Before you install Safeguard, ensure that the partitions that will contain `/opt/quest` have sufficient space available. • Sufficient space for the keystroke logs, application logs, and event logs. The size of this space depends on the number of servers, the number of commands, and the number of policies configured. • The space can be on a network disk drive rather than a local drive. • The server hosting Safeguard must be a separate machine dedicated to running the `pmmasterd` daemon.
SSH software	You must install and configure SSH client and server software on all policy server hosts. You must also install SSH client software on all hosts that will use the Sudo Plugin. You must enable access to SSH as the `root` user on the policy server hosts during configuration of the policy servers. Both OpenSSH 4.3 (and later) and Tectia SSH 6.4 (and later) are supported.
Processor	Policy Servers: 4 cores
RAM	Policy Servers: 8GB

Safeguard for Sudo Requirements

Table 2: Primary policy server and host system installation requirements

Systems Required	Minimum Requirements
Primary Policy Server	• Supported Unix or Linux operating system • SSH (ssh-keyscan binary)
Host System	• Supported Unix, Linux, or macOS platform • SSH (ssh-keyscan binary) • Sudo 1.8.1 (or later)

Default Ports

Configure the firewall ports appropriately when installing the Sudo Plugin on separate machines from the policy server.

Table 3: Masterport requirements

Variable	Default Port	Description
masterport	12345	TCP/IP port for `pmmasterd`. Safeguard uses the masterport to communicate with the `pmmasterd` (policy server daemon).

Supported platforms

The following table provides a list of supported platforms for Safeguard clients.

NOTE: Beginning with version 7.2 , Safeguard for Sudo supports only Linux-based systems for Safeguard policy servers.

Table 4: Linux supported platforms — server and plugin

Platform	Version	Architecture
Amazon Linux	AMI, 2	x86_64
CentOS Linux	6, 7, 8	Current Linux architectures: s390x, PPC64, PPC64le, x86, x86_64, AARCH64
Debian	Current supported releases	x86_64, x86, AARCH64
Fedora Linux	Current supported	x86_64, x86, AARCH64

Platform	Version	Architecture
	releases	
OpenSuSE	Current supported releases	x86_64, x86, AARCH64
Oracle Enterprise Linux (OEL)	6, 7, 8	Current Linux architectures: s390x, PPC64, PPC64le, x86, x86_64, AARCH64
Red Hat Enterprise Linux (RHEL)	6, 7, 8	Current Linux architectures: s390x, PPC64, PPC64le, x86, x86_64, AARCH64
SuSE Linux Enterprise Server (SLES)/Workstation	11 SP4, 12, 15	Current Linux architectures: s390x, PPC64, PPC64le, x86, x86_64, AARCH64
Ubuntu	Current supported releases	x86_64, x86, AARCH64

Table 5: Unix and Mac supported platforms — plugin

Platform	Version	Architecture
Apple MacOS	10.15 or later	x86_64, ARM64
FreeBSD	12.x, 13.x	x32, x64
HP-UX	11.31	PA, IA-64
IBM AIX	6.1 TL9, 7.1 TL3, 7.2	Power 4+
Oracle Solaris	10 8/11 (Update 10), 11.x	SPARC, x64

Reserve special user and group names

Reserve the following names for Safeguard usage:

- `pmpolicy` (user and group)
- `pmlog` (group)

For more information, see Reserve special user and group names on page 17..

Required privileges

You will need root privileges to install Safeguard software. Either log in as root or use the su program to acquire root privileges. Due to the importance of the root account, Safeguard carefully protects the system against certain accidental or deliberate situations that might lead to a breach in security. For example, if Safeguard discovers that its configuration files are open to modification by non-root users, it will reject all job requests. Furthermore, all Safeguard directories back to the / directory are checked for security in the same way, to guard against accidental or deliberate replacement.

Estimating size requirements

Policy server deployment requirements

The following recommendations are only provided as a rough guideline. The number of policy servers required for your environment may vary greatly depending on usage.

- One policy server is suitable for small test environments with less than 50 hosts.
- Production environments should have a minimum of two policy servers.
- Add an additional policy server for every 150-200 Safeguard hosts.
- Additional policy servers may be required to support geographically disparate locations.

Safeguard licensing

Safeguard 7.2 licensing options include:

30-day evaluation licenses

Safeguard for Sudo evaluation license allows you to manage unlimited Sudo Plugin hosts for 30 days; after 30 days, you are allowed to manage 10 Sudo Plugin hosts without receiving an alert.

A newly installed policy server comes with an evaluation license. You can install multiple evaluation licenses.

Commercial licenses

Both a **Sudo Policy** and a **Sudo Keystoke** license is required for Safeguard for Sudo features.

Although licenses are allocated on a per-agent basis, you install the licenses on Safeguard policy servers.

The `pmlicense` command allows you to display current license information, update a license (an expired one or a temporary one before it expires) or create a new one. See Installing licenses on page 39 or Displaying license usage on page 39 for more examples of using the `pmlicense` command.

Deployment scenarios

You can deploy Safeguard software within any organization using UNIX and/or Linux systems. Safeguard offers a scalable solution to meet the needs of the small business through to the extensive demands of the large or global organization.

There is no right or wrong way to deploy Safeguard, and an understanding of the flexibility and scope of the product will aid you in determining the most appropriate solution for your particular requirements. This section describes the following sample implementations:

- a single host installation
- a medium-sized business installation
- a large business installation

Configuration options

Decide which of the following configurations you want to set up:

1. **Primary Server Configuration**: Configure a single host as the primary policy server hosting the security policy for the policy group using either the pmpolicy (Privilege Manager for Unix) or sudo (Safeguard for Sudo) policy type. See Security policy types on page 43 for more information about these policy types.

   If you are configuring the primary policy server using the pmpolicy policy type, see the *One Identity Privilege Manager for Unix Administration Guide*.
2. **Secondary Server Configuration**: Configure a secondary policy server in the policy server group to obtain a copy of the security policy from the primary policy server.
3. **Sudo Plugin Configuration**: Join a Safeguard for Sudo host to a sudo policy or pmpolicy server group.

Single host deployment

A single-host installation is typically appropriate for evaluations, proof of concept, and demonstrations of Safeguard. This configuration example installs all of the components on a single UNIX/Linux host, with protection offered only within this single host. All logging and auditing takes place on this host.

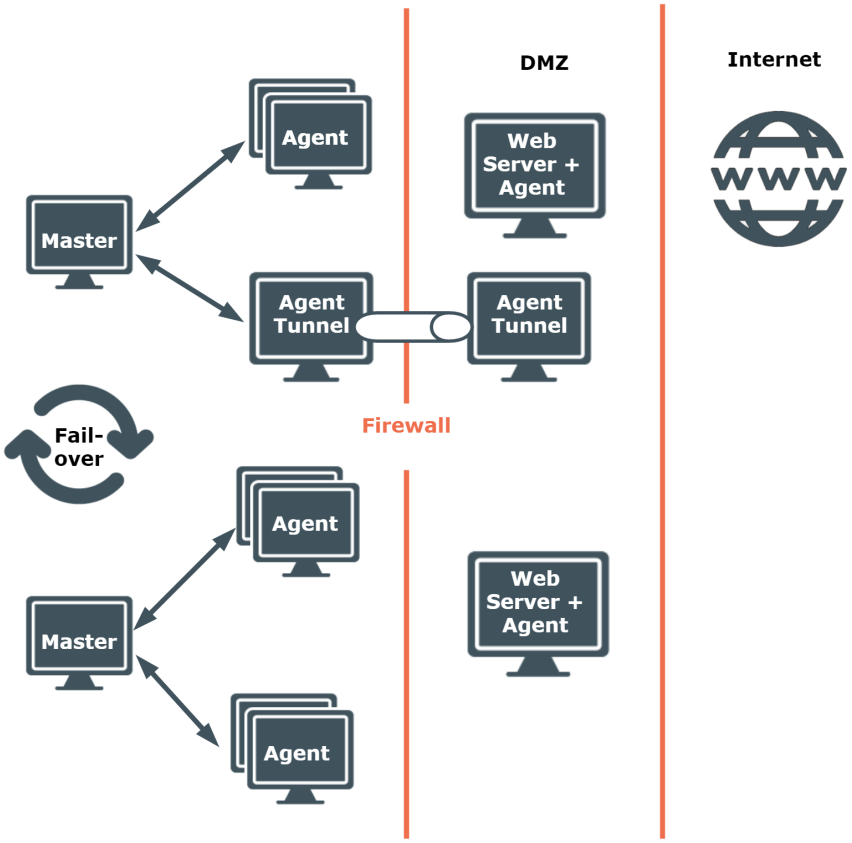
Medium business deployment

The medium business model is suitable for small organizations with relatively few hosts to protect, all of which may be located within a single data center.

This configuration example comprises multiple UNIX/Linux hosts located within the SME space and one or more web servers located in a DMZ.

Multiple policy server components (pmmasterd on page 162) are installed in a failover configuration, with groups of plugin hosts balanced between the policy servers. If a policy server is unavailable for any reason, the plugin hosts will failover to the alternative policy server.

Figure 3: Medium business implementation: Minimum 2 Masters and Circa 100 Agents

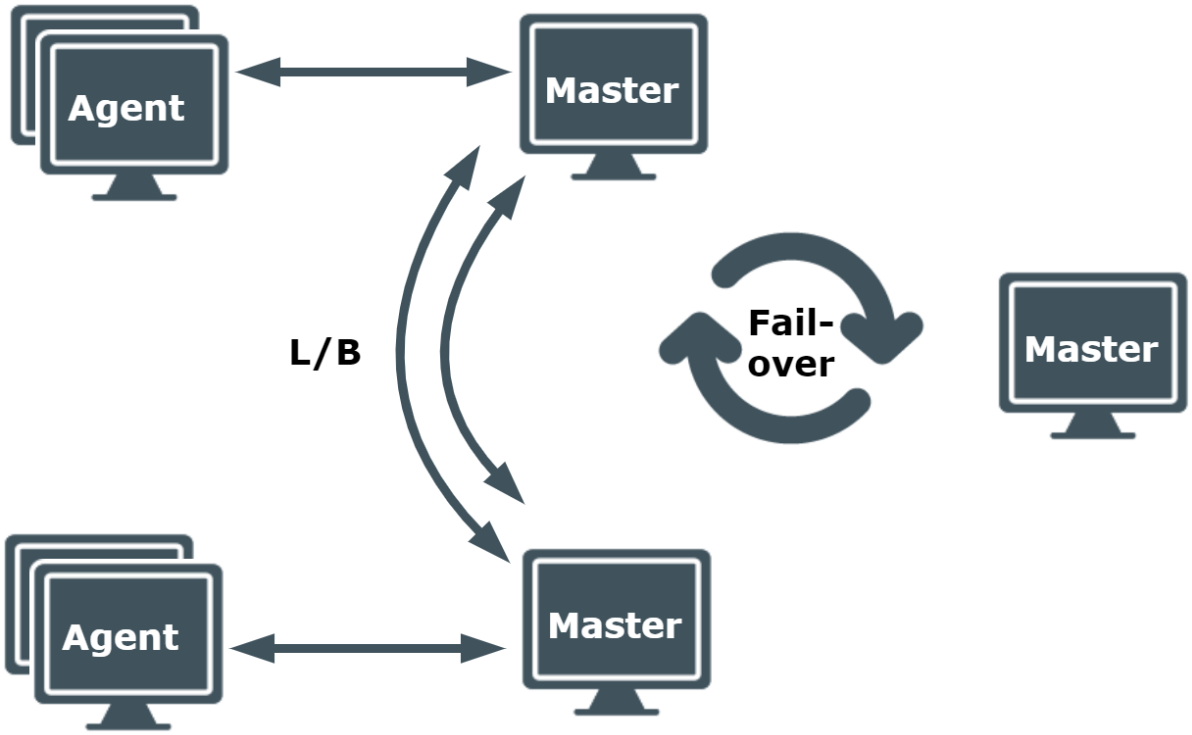


Large business deployment

This is an example of how a large business might deploy Safeguard. Some global companies prefer to fragment their requirement and deploy multiple instances as shown in the medium-sized business model.

This example comprises three policy servers, two are balancing the load of multiple plugin hosts. This may be necessary if there is a high level of audit and/or a significant volume of requested elevated privilege. Further, there is an additional policy server configured as a failover should one or both policy servers become unavailable.

Figure 4: Large business implementation: Minimum 3 Masters and less than 1000 Agents



Installation and Configuration

This is an overview of the steps necessary to set up your environment to use Safeguard software:

To configure a primary policy server

1. Check the server for installation readiness.
2. Install the Safeguard policy server package.
3. Configure the primary policy server.
4. Join the primary policy server to policy group.

To configure a secondary policy server

1. Check the host for installation readiness.
2. Install the Safeguard policy server package.
3. Configure the secondary policy server.
4. Join the Sudo Plugin host to the secondary policy server.

To install the Sudo Plugin on a remote host

1. Check the remote host for installation readiness.
2. Install the Safeguard software on the remote host.
3. Join the Sudo Plugin host to the policy server.

The following topics walk you through these steps.

Download Safeguard for Sudo software packages

To download the Safeguard for Sudo software packages

1. Go to https://support.oneidentity.com/safeguard-for-sudo.
2. On the **Product Support - Safeguard for Sudo** page, click **Software Downloads** under **Self Service Tools** in the left pane.
3. On the **Safeguard for Sudo - Download Software** page, click **Download** to the right of the version to be downloaded.

   See Installation Packages on page 189 for more information about Safeguard native platform install packages.
4. Read the License Agreement, select the **I have read and accept the agreement** option, and click **Submit**.
5. Download the relevant package from the web page.

   The Safeguard server package includes the PM Agent and the Sudo Plugin components.

Verifying package signature

All packages shipped by One Identity come with a signature. Signature verification depends on the platform:

- MacOS packages are signed by an Apple developer certificate.
- Linux, FreeBSD, AIX, Solaris and HP-UX packages are signed with a PGP key.

You can find the public key at pgp.mit.edu and at keyserver.ubuntu.com.

To fetch the public key, use its id:

```
gpg --keyserver <keyserver> --recv C5C4EC20AFB5B8E678085F81B161CD624417450C
```

You can also find the same public key in the `oneidentity_pgpkey.pub` file. To import it, use the following command:

```
gpg --import oneidentity_pgpkey.pub
```

To verify package signature

1. Download the public key.
2. Verify the files.

- For platforms with separate `.sig` file signatures, use `gpg2`:

  ```
  gpg --verify <file>.sig <file>
  ```

- For rpm packages, import the public key into the rpm's database:

  ```
  gpg --export -a "C5C4EC20AFB5B8E678085F81B161CD624417450C" >pubkey
  ```

  ```
  rpm --import pubkey
  ```

  And verify with:

  ```
  rpm --checksig --verbose <file>
  ```

- For debian packages, use `debsig-verify`.

Configure a Primary Policy Server

The first thing you must do is install and configure the host you want to use as your primary policy server.

Checking the server for installation readiness

Safeguard comes with a Preflight program that checks to see if your system meets the install requirements.

To check for installation readiness

1. Log on as the `root` user.
2. Change to the directory containing the `qpm-server` package for your specific platform.

   For example, on a 64-bit Red HatLinux, run:

   ```
   # cd server/linux-x86_64
   ```

3. To ensure that the `pmpreflight` command is executable, run:

   ```
   # chmod 755 pmpreflight
   ```

4. To verify your primary policy server host meets installation requirements, run:

   ```
   # sh pmpreflight.sh --server
   ```

   Running `pmpreflight.sh --server` performs these tests:

   - Basic Network Conditions:
     - Hostname is configured
     - Hostname can be resolved
     - Reverse lookup returns its own IP
   - Safeguard Server Network Requirements:
     - Policy server port is available (TCP/IP port 12345)
   - Safeguard Prerequisites:
     - SSH keyscan is available

5. Resolve any reported issues and rerun `pmpreflight` until all tests pass.

TCP/IP configuration

Safeguard uses TCP/IP to communicate with networked computers, so it is essential that you have TCP/IP correctly configured. If you cannot use programs such as `ssh` and `ping` to communicate between your computers, then TCP/IP is not working properly; consult your system administrator to find out why and make appropriate changes.

Ensure that your host has a statically assigned IP address and that your host name is not configured to the loopback IP address 127.0.0.1 in the `/etc/hosts` file.

Hosts database

Ensure that each host on your network knows the names and IP addresses of all other hosts. This information is stored either in the `/etc/hosts` file on each machine, or in NIS maps or DNS files on a server. Whichever you use, ensure all host names and IP addresses are up-to-date and available.

Safeguard components must be able to use forward and reverse lookup of the host names and IP addresses of other components.

Reserve special user and group names

It is important for you to reserve the following special user and group names for Safeguard usage:

- Users: `pmpolicy`, `pmclient`
- Groups: `pmpolicy`, `pmlog`

The `pmpolicy` user is created on a primary or secondary server. It is a non-privileged service account (that is, it does not require root-level permissions) that is used to synchronize the security policy on policy servers.

The `pmclient` user is created on a Sudo Plugin host. It is a non-privileged service account (that is, it does not require root-level permissions) that is used to synchronize the security policy on Sudo Plugin hosts (offline policy cache).

The `pmlog` and `pmpolicy` groups are used to control access to log files and the security policy, respectively.

Policy server daemon hosts

Safeguard requires that you choose a host to act as the policy server. This machine will run the `pmmasterd` daemon and must be available to manage requests for the whole network.

Run the policy server daemon on the most secure and reliable node. To maximize security, ensure the computer is physically inaccessible and carefully isolated from the network.

The policy server requires that the `pmmasterd` port (TCP/IP port 12345, by default) is available, and that Sudo Plugin hosts joined to the policy server are able to communicate with the policy server on this network port.

You can run multiple policy servers for redundancy and stability. Safeguard automatically selects an available policy server if more than one is on the network. For now, choose one machine to run `pmmasterd`. See pmmasterd on page 162 for more information.

Check Sudo version

Ensure that hosts running the Sudo Plugin have Sudo 1.8.1 (or later) installed.

If you have multiple instances of Sudo, update the PATH environment variable to ensure Safeguard for Sudo uses the correct version.

Installing the Safeguard packages

After you make sure your primary policy server host meets the system requirements, you are ready to install the Safeguard packages.

To install the Safeguard packages

1. From the command line of the host designated as your primary policy server, run the platform-specific installer. For example, run:

   ```
   # rpm --install qpm-server-*.rpm
   ```

   The Solaris server has a filename that starts with `QSFTpmsrv`.

When you install the `qpm-server` package, it installs all three Safeguard components on that host: the Safeguard Policy Server, the PM Agent, and the Sudo Plugin.

For details instructions on installing and configuring Privilege Manager for Unix, see the *One Identity Privilege Manager for Unix Administration Guide*.

Adding directories to PATH environment

After you install the primary policy server, you may want to update your PATH to include the Safeguard command.

To add quest-specific directories to your PATH environment

1. If you are a Safeguard administrator, add these quest-specific directories to your PATH environment:

```
/opt/quest/bin:/opt/quest/sbin
```

Configuring the Safeguard for Sudo Primary Policy Server

In Safeguard for Sudo, the policy server acts as a central sudoers policy store for all clients with the Sudo Plugin which have been joined to the policy group. The policy server also provides centralized event tracking and keystroke logging for the Sudo Plugin hosts.

The policy server also provides a revision management system, which allows tracking and reporting on changes made to the policy. If, for example, an important entry was accidentally removed from the sudoers file, you can restore a previous version of the policy.

The first policy server configured for a policy group is the primary policy server and holds the *master* copy of the policy. You configure a policy server by running the `pmsrvconfig` command without any options, like this:

```
# pmsrvconfig
```

`pmsrvconfig` runs with a set of default values and only prompts you when necessary.

To override the default values, you may specify a number of options. For more information about the various command options used in the following examples, see pmsrvconfig on page 184.

To configure a policy server for a sudo policy type

1. Run this command:

   ```
   # /opt/quest/sbin/pmsrvconfig
   ```

   By default, the local `/etc/sudoers` policy file is used and imported into the policy server repository. To import an alternate sudoers file, run the command with the `-f` option, as follows:

   ```
   # /opt/quest/sbin/pmsrvconfig -f <sudoers>
   ```

   where: `<sudoers>` is the path to the alternate sudoers file. For example:

   ```
   # /opt/quest/sbin/pmsrvconfig -f /tmp/sudoers
   ```

2. Accept the End User License Agreement (EULA) to configure the policy server.
3. When prompted, set the password for the new `pmpolicy` user.

   This password is also called the "Join" password. It is used to setup an SSH key between the sudo host and the server for the off-line policy caching feature. You are required to use this password when you add secondary policy servers or join remote hosts to this policy group.
4. (Optional) All Safeguard commands are in the `/opt/quest/sbin` and `/opt/quest/bin` directories, so you may want to update your PATH to include them, as follows:

   ```
   # PATH=$PATH:/opt/quest/sbin:/opt/quest/bin
   ```

   If you have multiple instances of sudo, updating the PATH environment variable ensures Safeguard for Sudo uses the correct version.

Configuring additional policies on a policy server

The sudo policy type supports multiple named policies in the policy server group. On the policy server, these named policies are represented as separate directories in the policy repository. Policy files are maintained using the pmpolicy command.

To configure additional policies on a policy server

1. To create a webservers policy from the file `/etc/sudoers.web`, run the following commands:

   ```
   # pmpolicy checkout -d policydir
   ```

   ```
   # mkdir policydir/policy_sudo/webservers
   ```

```
# cp /etc/sudoers.web policydir/policy_sudo/webservers/sudoers
```

```
# pmpolicy add -d policydir -p webservers/sudoers -n
```

The command checks out a copy of the current policy repository, creates a webserver directory for the new policy, populates it with the contents of the file /etc/sudoers.web, and commits the changes. After the policy directory is present on the server, a client can join to it.

Safeguard for Sudo Server Configuration Settings

The following table lists the default and alternative configuration settings when configuring a Safeguard for Sudo server. See PM settings variables on page 112 for more information about the policy server configuration settings.

Table 6: Safeguard for Sudo Server configuration settings

Configuration Setting	Default	Alternate
Configure Safeguard Policy Mode		
Policy mode: See Security policy types on page 43 for more information about policy types. Sets `policymode` in `pm.settings`. (Policy "modes" are the same as policy "types" in the console.)	sudo	The Sudo Plugin supports the **sudo** policy type and the **pmpolicy** type.
Configure host as primary or secondary policy group server:	primary	Enter **secondary**, then supply the primary server host name.
Policy Group Name: Sets `sudoersfile` in `pm.settings`.	<FQDN name of policy server>	Enter policy group name of your choice.
Path to sudoers file to import:	/etc/sudoers	Enter a path of your choice
Configure Safeguard Daemon Settings		
Policy server command line options:	-ar	Enter:

Configuration Setting	Default	Alternate
Sets `pmmasterdopts` in `pm.settings`.		• **-a** to send job acceptance messages to syslog. • **-e <logfile>** to use the error log file identified by <logfile>. • **-r** to send job rejection messages to syslog. • **-s** to send error messages to syslog. • **none** to assign no options. -a, -r, and -s override `syslog no` option; -e <logfile> overrides the `pmmasterdlog <logfile>` option.
Configure policy server host components to communicate with remote hosts through firewall?	No	Do not change this setting, because firewall options to not apply to the Sudo Plugin.
Define host services?	Yes Adds services entries to the `/etc/services` file.	Enter **No** You must add service entries to either the `/etc/services` file or the NIS services map.
Communications Settings for Safeguard		
Policy server daemon port number: Sets `masterport` in `pm.settings`.	12345	Enter a port number for the policy server to communicate with agents and clients.
Specify a range of reserved port numbers for this host to connect to other defined Safeguard hosts across a firewall? Sets `setreserveportrange` in `pm.settings`.	No	Enter **Yes**, then enter a value between 600 and 1023: 1. Minimum reserved port. (Default is 600.) 2. Maximum reserved port. (Default is 1023.)
Specify a range of non-reserved port numbers for this host to connect	No	Enter **Yes**, then enter a value between 1024 and 65535:

Configuration Setting	Default	Alternate
to other defined Safeguard hosts across a firewall? Sets `setnonreserveportrange` in `pm.settings`.		• Minimum non-reserved port. (Default is 1024.) • Maximum non-reserved port. (Default is 31024.)
Allow short host names? Sets `shortnames` in `pm.settings`.	Yes	Enter **No** to use fully-qualified host names instead.
Configure Kerberos on your network? Sets `kerberos` in `pm.settings`.	No	Enter **Yes**, then enter: 1. Policy server principal name. (Default is `host`.) 2. Local principal name. (Default is `host`.) 3. Directory for replay cache. (Default is `/var/tmp`. 4. Path for the Kerberos configuration files [`krbconf` setting]. (Default is `/etc/opt/quest/vas/vas.conf`.) 5. Full pathname of the Kerberos keytab file [`keytab` setting]. (Default is `/etc/opt/quest/vas/host.keytab`.
Encryption level: Sets `encryption` in `pm.settings`.	AES	Enter one of these encryption options: • DES • TRIPLEDES • AES
Enable certificates? Sets `certificates` in `pm.settings`.	No	Enter **Yes**, then answer: Generate a certificate on this host? (Default is NO.) Enter **Yes** and specify a passphrase for the certificate.

Configuration Setting	Default	Alternate
		Once configuration of this host is complete, swap and install keys for each host in your system that need to communicate with this host. See Swap and install keys on page 26 for details.
Activate the failover timeout?	No	Enter **Yes**, then assign the failover timeout in seconds: (Default is 10.)
Failover timeout in seconds Sets `failovertimeout` in `pm.settings`.	10	Enter timeout interval.
Configure Safeguard Logging Settings		
Send errors reported by the policy server and local daemons to syslog?	Yes	Enter **No**
Policy server log location: Sets `pmmasterdlog` in `pm.settings`.	/var/-log/pmmasterd.log	Enter a location.
Configure Safeguard Sudo Plugin		
Configure Sudo Plugin?	No	Enter **Yes**
Install Safeguard Licenses		
XML license file to apply:	(use the freeware product license)	Enter the location of the .xml license file. Enter **Done** when finished.
Enter <password> This password is also called the "Join" password. You will use this password when you add secondary policy servers or join remote hosts to this policy group.		

You can find an installation log file at: `/opt/quest/qpm4u/install/pmsrvconfig_output_<Date>.log`

Join hosts to policy group

Once you have installed and configured the primary policy server, you are ready to join it to a policy group. When you join a policy server to a policy group, it enables that host to validate security privileges against a single common policy file located on the primary policy server, instead of on the host.

For Sudo Plugin hosts (`qpm-plugin`), you must "join" your policy servers to the policy groups using the `pmjoin_plugin` command.

Joining Sudo Plugin to Policy Server

Run the `pmjoin_plugin` command after installing the Sudo Plugin package (`qpm-plugin`) on a remote host to allow it to communicate with the servers in the policy group.

To join Sudo Plugin to policy server

1. Join the Sudo Plugin host to the policy server by running the following command:

   ```
   # pmjoin_plugin <primary_policy_server>
   ```

   where `<primary_policy_server>` is the host name of the primary policy server.
2. To automatically accept the End User License Agreement (EULA), use the `-a` option with the "join" command, as follows:

   ```
   # pmjoin_plugin -a <primary_policy_server>
   ```

You have now joined the host to a primary policy server. The primary policy server is now ready to accept commands using sudo.

Joining Sudo Plugin to policy server using a non-default policy

When joining a policy group, the client may specify a policy name to use a policy other than the default sudoers file.

To join Sudo Plugin to policy server using a non-default policy

1. Join a client to the webservers policy mentioned above by running the following command:

   ```
   pmjoin_plugin -N webservers <primary_policy_server>
   ```

   If the named policy does not exist on the server, the client will be unable to join.

Swap and install keys

If certificates are enabled in the `/etc/opt/quest/qpm4u/pm.settings` file of the primary server, then you must exchange keys (swap certificates) prior to joining a client or secondary server to the primary server. Optionally, you can run the configuration or join with the `-i` option to interactively join and exchange keys.

One Identity recommends that you enable certificates for higher security.

The examples below use the keyfile paths that are created when using interactive configuration or join if certificates are enabled.

To swap certificate keys

1. Copy Host2's key to Host1. For example:

   ```
   # scp /etc/opt/quest/qpm4u/.qpm4u/.keyfiles/key_localhost \
   root@Host1:/etc/opt/quest/qpm4u/.qpm4u/.keyfiles/key_server2
   ```

2. Copy Host1's certificate to Host2. For example:

   ```
   # scp root@host1:/etc/opt/quest/qpm4u/.qpm4u/.keyfiles/key_localhost \
   /etc/opt/quest/qpm4u/.qpm4u/.keyfiles/key_host1
   ```

3. Install Host1's certificate on Host2. For example:

   ```
   # /opt/quest/sbin/pmkey -i /etc/opt/quest/qpm4u/.qpm4u/.keyfiles/key_host1
   ```

4. Log on to Host1 and install Host2's certificate. For example:

   ```
   # /opt/quest/sbin/pmkey -i /etc/opt/quest/qpm4u/.qpm4u/.keyfiles/key_host2
   ```

If you use the interactive configure or join, the script will exchange and install keyfiles automatically.

Configure a secondary policy server

The *primary* policy server is always the first server configured in the policy server group; *secondary* servers are subsequent policy servers set up in the policy server group to help with load balancing. The "master" copy of the policy is kept on the primary policy server.

All policy servers (primary and secondary) maintain a production copy of the security policy stored locally. The initial production copy is initialized by means of a `checkout` from the repository when you configure the policy server. Following this, the policy servers automatically retrieve updates as required.

By adding one or more secondary policy servers, the work of validating policy is balanced across all of the policy servers in the group, and provides failover in the event a policy

server becomes unavailable. Use `pmsrvconfig` with the `-s` option to configure the policy server as a secondary server.

Installing secondary servers

To install the secondary server

1. From the command line of the host designated as your secondary policy server, log on as the `root` user.
2. Change to the directory containing the `qpm-server` package for your specific platform.

   For example, on a 64-bit Red Hat Linux, run:

   ```
   # cd server/linux-x86_64
   ```

3. Run the platform-specific installer. For example, run:

   ```
   # rpm --install qpm-server-*.rpm
   ```

   The Solaris server has a filename that starts with `QSFTpmsrv`.

   When you install the `qpm-server` package, it installs all three Safeguard components on that host:

   - Safeguard Policy Server
   - PM Agent (which is used by Privilege Manager for Unix)
   - Sudo Plugin (which is used by Safeguard for Sudo)

   You can only join a PM Agent host to a Safeguard policy server or a Sudo Plugin host to a sudo policy server. See Security policy types on page 43 for more information about policy types.

Configuring a secondary server

You use the `pmsrvconfig -s <primary_policy_server>` command to configure a secondary server. See pmsrvconfig on page 184 for more information about the `pmsrvconfig` command options.

To configure the secondary server

1. From the command line of the secondary server host, run:

   ```
   # pmsrvconfig -s <primary_policy_server>
   ```

   where `<primary_policy_server>` is the hostname of your primary policy server.

`pmsrvconfig` prompts you for the "Join" password from the primary policy server, exchanges ssh keys for the pmpolicy service user, and updates the new secondary policy server with a copy of the *master* (production) policy.

Once you have installed and configured a secondary server, you are ready to join the Sudo Plugin to it. See Join hosts to policy group on page 25 for details.

Synchronizing policy servers within a group

Safeguard generates log files containing event timestamps based on the local clock of the authorizing policy server.

To synchronize all policy servers in the policy group, use Network Time Protocol (NTP) or a similar method of your choice.

Install Sudo Plugin on a remote host

Once you have installed and configured the primary policy server, you are ready to install a Sudo Plugin on a remote host.

Checking Sudo Plugin Host for installation readiness

To check a Sudo Plugin host for installation readiness

1. Log on to the remote host system as the `root` user and navigate to the files you extracted on the primary policy server.
2. From the root directory, run a readiness check to verify the host meets the requirements for installing and using the Sudo Plugin, by running:

   ```
   # sh pmpreflight.sh --sudo --policyserver <myhost>
   ```

   where `<myhost>` is the hostname of the primary policy server.

   Running `pmpreflight.sh --sudo` performs these tests:

   - Basic Network Conditions:
     - Hostname is configured
     - Hostname can be resolved
     - Reverse lookup returns it own IP

- Policy Server Connectivity
  - Hostname of policy server can be resolved
  - Can ping the policy server
  - Can make a connection to policy server
  - Policy server is eligible for a join
- Sudo Installation
  - sudo is present on the host
  - sudo is in a functional state
  - sudo is version 1.8.1 (or later)
- Prerequisites to support off-line policy caching
  - SSH keyscan is available
  - Policy server port is available

3. Resolve any reported issues and rerun `pmpreflight` until all tests pass.

Installing a Sudo Plugin on a remote host

To install a Sudo Plugin on a remote host

1. Log on as the `root` user.
2. Change to the directory containing the `qpm-plugin` package for your specific platform. For example, on a 64-bit Red Hat Linux, enter:

   ```
   # cd sudo_plugin/linux-x86_64
   ```

3. Run the platform-specific installer. For example, on Red Hat Linux run:

   ```
   # rpm --install qpm-plugin-*.rpm
   ```

Once you install the Sudo Plugin package, the next task is to join it to the policy server.

Joining a Sudo Plugin to a primary policy server

Once you have installed a Sudo Plugin on a remote host you are ready to join it to the primary policy server. Joining a host to a policy server enables it to communicate with the servers in the policy group.

The `pmjoin` command configures PM Agents (`qpm-agent` package) while the `pmjoin_plugin` command configures Sudo Plugin hosts (`qpm-plugin` package).

To join a Sudo Plugin to the primary policy server

1. Run the following command:

   ```
   # pmjoin_plugin <primary_policy_server>
   ```

   where `<primary_policy_server>` is the host name of the primary policy server.

   To automatically accept the End User License Agreement (EULA), use the `-a` option with the "join" command, as follows:

   ```
   # pmjoin_plugin -a <primary_policy_server>
   ```

When you join a Sudo Plugin to a policy server, Safeguard for Sudo adds the following lines to the current local sudoers file, generally found in `/etc/sudoers`.

```
##
 ## WARNING: Sudoers rules are being managed by Safeguard for Sudo
 ## WARNING: Do not edit this file, it is no longer used.
 ##
 ## Run "/opt/quest/sbin/pmpolicy edit" to edit the actual sudoers rules.
 ##
```

When you unjoin the Sudo Plugin, Safeguard for Sudo removes those lines from the local sudoers file.

You have now installed the Safeguard for Sudo packages, configured a primary policy server for the sudo policy type, and joined the Sudo Plugin to the primary policy server. The primary policy server is ready to accept commands using sudo.

Verifying Sudo Plugin configuration

If you have installed the Sudo Plugin component using the `qpm-plugin` package, use the `pmplugininfo` command to verify the plugin configuration.

To verify the Sudo Plugin configuration

1. From the command line, run:

   ```
   # pmplugininfo
   ```

   The `pmcplugininfo` command displays the current configuration settings. For example:

```
[0][root@host2 /]# pmplugininfo
   - Joined to a policy group                  : YES
   - Name of policy group                      : polsrv1.example.com
   - Hostname of primary policy server         : polsrv1.example.com
   - Policy type configured on policy group    : sudo
   - Pathname of compatible sudo binary        : /usr/local/bin/sudo v1.8.2
[0][root@host2 /]#
```

The secondary server Sudo Plugin will be joined to the secondary server. This is unique because all other Sudo Plugin hosts must join to the primary server.

Load balancing on the client

Load balancing is handled on each client, using information that is returned from the policy server each time a session is established.

If a session cannot be established because the policy server is unavailable (or offline) that policy server is marked as *unavailable*, and no further sudo sessions are sent to it until the next retry interval.

`pmpluginloadcheck` runs transparently on each host to check the availability and loading of the policy server. When a policy server is marked as *unavailable*, `pmpluginloadcheck` attempts to connect to it at intervals. If it succeeds, the policy server is marked as *available* and able to run Safeguard sessions.

To view the current status of the policy server

- Run the following command:

```
# pmpluginloadcheck [-f]
```

If the policy server cannot be contacted, the last known information for this host is reported.

Remove configurations

You can remove the Safeguard Server or Sudo Plugin configurations by using the `-u` option with the following commands:

- `pmsrvconfig` to remove the Safeguard Server configuration
- `pmjoin_plugin` to remove the Sudo Plugin configuration

Take care when you remove the configuration from a policy server, particularly if the policy server is a primary server with secondary policy servers in the policy group, as agents joined to the policy group will be affected.

Uninstalling the Safeguard software packages

To uninstall the Safeguard packages

1. Log in and open a root shell.
2. Use the package manager for your operating system to remove the packages:

Table 7: Safeguard Server uninstall commands

Package	Command
RPM	# rpm -e qpm-server
DEB	# dpkg -r qpm-server

Table 8: Sudo Plugin uninstall commands

Package	Command
RPM	# rpm -e qpm-plugin
DEB	# dpkg -r qpm-plugin
Solaris	# pkgrm QSFTpmplg
HP-UX	# swremove qpm-plugin
AIX	# installp -u qpm-plugin

Uninstalling Safeguard for Sudo on macOS

To uninstall Safeguard for Sudo on macOS

1. Open `/Applications` and click **QPM4Sudo_Uninstaller**.
2. When it asks if you want to delete the `pmclient` user, enter **yes** or **no**.

To uninstall Safeguard for Sudo from the command line

1. Enter:

   ```
   /opt/quest/qpm4u/bin/uninstall
   ```

2. To delete the `pmclient` user, enter:

   ```
   dscl . -delete /Users/pmclient
   ```

5

Upgrade Safeguard for Sudo

Safeguard for Sudo supports a direct upgrade installation from version 2.0. The Safeguard software in this release is provided using platform-specific installation packages.

If you are currently running Privilege Manager for Unix 2.0, it may be possible to perform a direct upgrade installation depending on the package management software on your platform (Note: Direct upgrade installations are not possible with Solaris `.pkg` packages). If you perform a direct upgrade installation, your previous configuration details are retained. Where a direct upgrade is not possible, you must first remove the previously installed package, and install and configure Safeguard as a new product installation.

Before you upgrade

Because the Safeguard 7.2 original platform installer packages do not provide an automated rollback script, One Identity highly recommends that you back up important data such as your license, `pm.settings` file, policy, and log files before you attempt to upgrade your existing Safeguard policy servers.

To install Safeguard 7.2 , change to the directory where the install package is located for your platform and run the package installer. See Installing the Safeguard packages on page 18 for details about how to install the Safeguard for Sudo software.

Upgrading Safeguard packages

Safeguard has the following three packages:

- Server (qpm-server)
- PM Agent (qpm-agent) - Used by Privilege Manager for Unix only
- Sudo Plugin (qpm-plugin) - Used by Safeguard for Sudo only

These packages are mutually exclusive, that is, you can only install one of these packages on a host at any given time.

For more information on installing/upgrading the PM Agent, see the *One Identity Privilege Manager for Unix Administration Guide*.

Upgrading the server package

To upgrade the server package

1. Change to the directory containing the `qpm-server` package for your specific platform. For example, on a 64-bit Red Hat Linux system, run:

   ```
   # cd server/linux-x86_64
   ```

2. Run the platform-specific installer. For example, run:

   ```
   # rpm --upgrade qpm-server-*.rpm
   ```

Upgrading the Sudo Plugin package

To upgrade the Sudo Plugin package

1. Change to the directory containing the `qpm-plugin` package for your specific platform. For example, on a 64-bit Red Hat Linux 5 system, run:

   ```
   # cd agent/linux-x86_64
   ```

2. Run the platform-specific installer. For example, run:

   ```
   # rpm --upgrade qpm-plugin*.rpm
   ```

Removing Safeguard packages

Where a direct upgrade is not possible, you must first remove the previously installed package, and install and configure Safeguard as a new product installation.

Removing the server package

To remove the server package

1. Run the package uninstall command for your operating system.

   For example, to remove the `qpm-server` package on a 64-bit Red Hat Enterprise Linux 5 system, run:

   ```
   # rpm --erase qpm-server
   ```

2. To complete the removal of the `qpm-server` package, delete:
   - `pmpolicy` service user
   - `pmpolicy` group
   - `pmlog` group
   - policy repository directories in `/etc/opt/quest/qpm4u/`

Removing the Sudo Plugin package

To remove the Sudo Plugin package

1. Run the package uninstall command for your operating system.

   For example, to remove the `qpm-plugin` package on a 64-bit Red Hat Enterprise Linux 5 system, run:

   ```
   # rpm --erase qpm-plugin
   ```

6

System Administration

Safeguard provides command line utilities to help you manage your policy servers. They can be used to check the status of your policy servers, edit the policy, or to simply report the information.

Reporting basic policy server configuration information

To report basic information about the configuration of a policy server

1. From the command line, enter:

   ```
   # pmsrvinfo
   ```

   This command returns output similar to this:

   ```
   Policy Server Configuration:
   ---------------------------
   Safeguard version            : 6.1.0 (nnn)
   Listening port for pmmasterd daemon  : 12345
   Comms failover method                : random
   Comms timeout(in seconds)            : 10
   Policy type in use                   : sudo
   Group ownership of logs              : pmlog
   Group ownership of policy repository : pmpolicy
   Policy server type                   : primary
   Primary policy server for this group : myhost.example.com
   Group name for this group            : MyPolicyGroup
   Location of the repository           : file:

   ////var/opt/quest/qpm4u/.qpm4u/.repository/sudo_repos/trunk
   Hosts in the group                   : myhost.example.com
   ```

Checking the status of the master policy

The "master" copy of the policy file resides in a repository on the primary policy server. Each primary and secondary policy server maintains a "production" copy of the policy file or files. Use the `pmpolicy` utility to verify that the production copy is current with the master policy.

To compare the production policy file against the master policy on the primary server

1. From the command line, enter:

   ```
   # pmpolicy masterstatus
   ```

   If the files are in sync, the `Current Revision` number will match the `Latest Trunk Revision` number. If someone hand-edited the local copy without using `pmpolicy` utility commands to commit the changes, "Locally modified" will indicate "YES".

   If the production policy is not current with the master policy you can update the production policy with `pmpolicy sync`.

Related Topics

pmpolicy

Checking the policy server

When the policy server is not working as expected, use the `pmsrvcheck` command to determine the state of the server and its configuration.

To verify the policy server is running

1. From the command line, enter:

   ```
   # pmsrvcheck
   ```

   This command returns output similar to this:

   ```
   testing policy server [ Pass ]
   ```

   If the policy server is working properly, the output returns '`pass`', otherwise it returns, '`fail`'.

Related Topics

pmsrvcheck

Checking policy server status

The primary and secondary policy servers need to communicate with each other. The Sudo Plugin hosts also need to communicate with the policy servers in the policy group. Run `pmpluginloadcheck` on the remote hosts to verify that they can communicate with the policy servers in the group.

To determine if there any issues with policy servers in the policy group

From the Safeguard for Sudo host command line, enter:

```
# pmpluginloadcheck -r
```

This command has output similar to this:

```
[0][root@sol10-x86 /]# pmpluginloadcheck -r
** Reporting current availability of each configured master...
   * Host:myhost1.example.com (172.16.1.129) ... [ OK ]
** Based on this data, the server list is currently ordered as:
1.     myhosts.example.com
```

Related Topics

pmpluginloadcheck

Checking the Sudo Plugin configuration status

To check the Sudo Plugin configuration status

1. From the command line, enter:

   ```
   # pmpluginfo
   ```

   This command returns output similar to this:

   ```
   # pmpluginfo
      - Joined to a policy group          : YES
      - Name of policy group              : MyPolicyGroup
      - Hostname of primary policy server : myhost.example.com
   ```

   If the Sudo Plugin has been properly configured, it will say `Joined to a Policy Group: YES` and give the policy group name and primary policy server's hostname.

Related Topics

pmpluginfo

Installing licenses

To install a license file

1. Copy the `.dlv` license file to the policy server.
2. To install the license, run:

   ```
   # /opt/quest/sbin/pmlicense -l <license_file>
   ```

   This command displays your currently installed license and the details of the new license to be installed.
3. When it asks, "`Would you like to install the new license (Y/N) [Y]?`", press **Enter**, or type: **Y**
4. If there are other policy servers configured in your policy server group, it forwards the license configuration to the other servers.

Related Topics

pmlicense

Displaying license usage

Use the `pmlicense` command to display how many client licenses are installed on the policy server on which you run the command.

Use `pmlicense` without any arguments to show an overall status summary, including the number of licenses configured and the total licenses in use for each license option.

To display current license status information

1. At the command line, enter:

   ```
   # pmlicense
   ```

   Safeguard displays the current license information, noting the status of the license. Your output will be similar to the following:

```
*** One Identity Safeguard ***
*** QPM4U VERSION
   7.2
  .0 (0xx) ***
*** CHECKING LICENSE ON HOSTNAME:user123.example.com, IP
ADDRESS:10.10.178.123 ***
*** SUMMARY OF ALL LICENSES CURRENTLY INSTALLED ***
   * License Type PERMANENT
   * Commercial/Freeware License COMMERCIAL
   * Expiration Date NEVER
   * Max QPM4U Client Licenses 0
   * Max Sudo Policy Plugin Licenses 10
   * Max Sudo Keystroke Plugin Licenses 0
   * Authorization Policy Type permitted ALL
   * Total QPM4U Client Licenses In Use 0
   * Total Sudo Policy Plugins Licenses In Use 4
   * Total Sudo Keystroke Plugins Licenses In Use 0
```

The above example shows that the current license allows for ten Sudo Policy Plugins (Sudo Plugin licenses) and four licenses are currently in use.

Use `pmlicense` with the `-us` option to view a summary usage report; use `-uf` to view the full usage report.

To show a full usage report including last use dates

1. At the command line, enter:

   ```
   # pmlicense -uf
   ```

   Your output will be similar to the following:

   ```
   Detailed Licensed Hosts Report
   ------------------------------------------------------------------------
   Number | Last Access Time                          | Hostname
   ------------------------------------------------------------------------
          | QPM4U | SudoPolicy         | SudoKeystroke |
   ------------------------------------------------------------------------
   1      |       |  2012/07/01 17:14  |               | admin1.example.com
   2      |       |  2012/07/01 17:14  |               | user101.example.com
   3      |       |  2012/07/01 16:28  |               | user123.example.com
   4      |       |  2012/07/01 17:14  |               | dev023.example.com
   ```

The above output shows the full report, including the host names and dates the Sudo Plugins used the policy server.

The `pmlicense` command supports many other command-line options.

Related Topics

pmlicense

Listing policy file revisions

After you have made several revisions to your policy file under source control, you can view the list of policy file versions stored in the repository.

To display all previous version numbers with timestamps and commit logs

1. From the command line, enter:

   ```
   # pmpolicy log
   ```

   This command returns output similar to this:

   ```
   ** Validate options                [ OK ]
    ** Check out working copy       [ OK ]
    ** Retrieve revision details [ OK ]
    version="3",user="pmpolicy",date=2011-05-11,time=19:27:01,msg=""
    version="2",user="pmpolicy",date=2011-05-11,time=19:19:47,msg="added
   tuser"
    version="1",user="pmpolicy",date=2011-05-11,time=15:56:12,msg="First
   import"
   ```

Viewing differences between revisions

You can view the changes from revision to revision of a policy file.

To show the differences between version 1 and version 3

1. From the command line, enter:

   ```
   # pmpolicy diff -r:1:2
   ```

   This command returns output similar to this:

   ```
   ** Validate options                                          [ OK ]
   ** Check out working copy (trunk revision)                   [ OK ]
   ** Check differences                                         [ OK ]
   ** Report differences between selected revisions             [ OK ]
      - Differences were detected between the selected versions
   Details:
   Index: sudoers
   ===================================================================
   --- sudoers (revision 1)
   +++ sudoers (revision 2)
   ```

```
@@ -88,6 +88,6 @@
# Defaults targetpw # Ask for the password of the target user
# ALL ALL=(ALL) ALL # WARNING: only use this together with 'Defaults
targetpw'

-## Read drop-in files from /etc/sudoers.d
+## Read drop-in files from sudoers.d
## (the '#' here does not indicate a comment)
-#includedir /etc/sudoers.d
+# includedir sudoers.d
```

The output reports lines removed and lines added in a unified diff format.

Backup and recovery

It is important for you to perform systematic backups of the following directories on all policy servers:

- `/var/opt/quest/qpm4u` which contains:
  - Event Logs
  - Keystroke Logs (I/O logs)
  - SVN Repository
  - SSH Keys
  - pmpolicy
- `/etc/opt/quest/qpm4u` which contains:
  - Settings File
  - Production Policy
- `/opt/quest/qpm4u/.license*` which contains:
  - License Files
- `/opt/quest/qpm4u/license*` which contains:
  - License Files
- `/opt/quest/qpm4u/install` which contains:
  - Install Logs
  - End User License Agreement (EULA)

When recovering from a failure, keep the same hostname and IP address.

7

Managing Security Policy

The Safeguard security system consists of one or more centralized policy servers and one or more remote clients. A user wishing to run a command secured by Safeguard makes a request to their client. The request is then propagated to the policy server which consults a security policy to determine whether to allow or disallow the command. A typical Safeguard installation has several policy servers to provide adequate fail-over and load-balancing coverage.

The Safeguard policy servers are capable of recording all the activity which passes through them. The power to accurately log `root`, and other account activities in a safe environment allows you to implement a secure system administration regime with an indelible audit trail. You always know exactly what is happening in `root`, as well as who did it, when it happened, and where.

The data created by the Safeguard policy servers is stored in a log file called an event log. An entry in the event log is made every time a policy server is used to run a command.

Security policy types

The security policy lies at the heart of Safeguard. Safeguard guards access to privileged functions on your systems according to rules specified in the security policy. It stipulates which users may access which commands with escalated privileges.

Safeguard supports two security policy types (or modes):

- **sudo policy type**: Safeguard for Sudo uses a standard sudoers file as its security policy; that is, the sudo policy is defined by the `sudoers` file which contains a list of rules that control the behavior of sudo. The `sudo` command allows users to get elevated access to commands even if they do not have `root` access.

  Safeguard uses the sudo policy type by default. The sudo policy type is only supported with the One Identity Safeguard for Sudo product.
- **pmpolicy type**: Privilege Manager for Unix uses an advanced security policy which employs a high-level scripting language to specify access to commands based on a wide variety of constraints. The Privilege Manager for Unix policy is defined in `pm.conf`, the default policy configuration file which contains statements and declarations in a language specifically designed to express policies concerning the

use of root and other controlled accounts.

Beginning with release 7.0, both Privilege Manager for Unix and Safeguard for Sudo support the pmpolicy type.

By default, the policy server configuration tool (pmsrvconfig) uses the sudo policy type on new installations; if you want to run Safeguard for Sudo using the pmpolicy type you must specify that explicitly when using the policy server configuration script.

The pmsrvconfig program is used by both Privilege Manager for Unix and Safeguard for Sudo. Run pmsrvconfig -m sudo or pmsrvconfig -m pmpolicy to specify the policy type. See pmsrvconfig on page 184 for more information about the pmsrvconfig command options.

The default behavior for setting up the initial policy depends on which type of policy you are using. If you configure Safeguard for Sudo using the default sudo policy type, pmsrvconfig uses a copy of the /etc/sudoers file as its initial security policy if the file exists, otherwise it creates a generic sudoers file.

When you join a Sudo Plugin to a policy server, Safeguard for Sudo adds the following lines to the current local sudoers file, generally found in /etc/sudoers.

```
##
## WARNING: Sudoers rules are being managed by Safeguard for Sudo
## WARNING: Do not edit this file, it is no longer used.
##
## Run "/opt/quest/sbin/pmpolicy edit" to edit the actual sudoers rules.
##
```

When you unjoin the Sudo Plugin, Safeguard for Sudo removes those lines from the local sudoers file.

Use the pmsrvconfig -f <path> command to override the default and import the initial security policy from the specified location. When using the sudo policy type, you can only use the -f option to import a file; you can not import a directory.

Safeguard uses a version control system to manage and maintain the security policy. This allows auditors and system administrators to track changes that have been made to the policy and also allows a single policy to be shared and distributed among several policy servers. The "master" copy of the security policy and all version information is kept in a repository on the primary policy server.

You manage the security policy using the pmpolicy command and a number of pmpolicy subcommands. It is important that you only make changes to the policy using the pmpolicy command. Using pmpolicy ensures that the policy is updated in the repository and across all policy servers in the policy group. You can run the pmpolicy command from any policy server in the policy group.

Do not edit the security policy on a policy server directly. Changes made using visudo will eventually be overwritten by the version control system.

The primary policy server uses a local service account, pmpolicy, to own and manage the security policy repository. The pmpolicy service account is set when you configure the primary policy server. At that time you assign the pmpolicy service account a password and set its home directory to /var/opt/quest/qpm4u/pmpolicy. This password is also called the "Join" password because you use it when you add secondary policy servers or join remote hosts to this policy group.

You can manually create the `pmpolicy` user prior to running the `pmsrvconfig` script, but if the user account does not exist, the script creates the user and asks you for a password.

When you run the `pmsrvconfig` command, it attempts to initialize the security policy by reusing an existing policy file on this host. If a security policy does not exist, it generates a default policy.

Specifying security policy type

To configure a Safeguard for Sudo policy server, you must specify the sudo policy type.

To specify the security policy type

1. To specify the sudo policy type, run:

   ```
   # pmsrvconfig -m sudo
   ```

2. To specify the pmpolicy type, run:

   ```
   # pmsrvconfig -m pmpolicy
   ```

   For more information about pmpolicy language, see *Privilege Manager for Unix Administration Guide*.

Related Topics

pmsrvconfig

The sudo type policy

A sudo type policy is used with the Safeguard for Sudo product. When you configure the primary policy server, if `/etc/sudoers` exists, it imports this file and uses it as the initial `sudoers` policy file. Otherwise, it creates a generic `sudoers` file.

By default, the Safeguard for Sudo `sudoers` file resides in `/etc/opt/quest/qpm4u/policy/sudoers`, but is not meant to be accessed directly.

Sudo type policy rules

Sudo type policy rules look like this:

```
Defaults   secure_path = /sbin:/bin:/usr/sbin:/usr/bin
root       ALL=(ALL) ALL
%wheel     ALL=(ALL) ALL
```

TIP: Sudo processes rules from top to bottom. This means that the order of rules is important because the last rule takes precedence. For this reason, always place exceptions under the generic settings.

Sudo rules are displayed in columns, denoting:

- which user or group
- on which server
- as which user
- has permission to run which command(s).

Different columns can be replaced with lists, and by using an alias, you can use a list in multiple places. The reserved word `ALL` is a built-in alias, that, when used in a command context, allows any user to run any command.

- Lines starting with `Defaults` are generic rules that change the default behavior of sudo. The line starting with `Defaults` in this example is a rule that applies to all users (but it can be limited to a subset of users).
- Lines starting with a user name (or a list of user names) are rules granting permissions to specific users. The line starting with `root` in this example is a rule granting permission to the root user to run any command on any host as any user.
- Lines starting with a % mark and a group name are rules granting permissions to a specific group. The line starting with `%wheel` in this example is a rule granting permission to all users in the wheel group to run any command on any host as any user.

  NOTE: You must prepend a percentage sign (%) in front of groups, but not in front of users.

In short, the example will let the root or any user in the wheel group run any command on any host as any user. However, even in this case, using sudo is beneficial because you can allow certain users to run commands as root without sharing the root password, while by logging all commands and arguments, sudo also provides an audit trail of the users doing so.

NOTE: Safeguard for Sudo does not use the `/etc/sudo.conf` file to load modules. Safeguard for Sudo uses the `sudoers` policy file and it uses a slightly different syntax. For more information, see Configuring a sudo approval plugin and Configuring a sudo audit plugin.

For more information on Defaults, aliases, or the sudoers syntax in general, see the *Sudoers man page*.

Viewing the security profile changes

To view a summary of the changes you made to your security policy

1. At the command line, run:

```
# pmpolicy log
```

```
** Validate options                [ OK ]
 ** Check out working copy       [ OK ]
 ** Retrieve revision details  [ OK ]
 version="3",user="pmpolicy",date=2012-07-11,time=15:43:30,msg="add
sudoers.d/helpdesk  "
 version="2",user="pmpolicy",date=2012-07-11,time=15:38:21,msg="add
#includedir  sudoers.d"
 version="1",user="pmpolicy",date=2012-07-11,time=15:35:19,msg="First
import"
```

2. To examine the differences between two versions, run:

```
# pmpolicy diff -r1:2
```

```
** Validate options                                              [ OK ]
** Check out working copy (trunk revision)                      [ OK ]
** Check differences                                            [ OK ]
** Report differences between selected revisions                [ OK ]
   - Differences were detected between the selected versions
Details:
Index: sudoers
===================================================================
--- sudoers (revision 1)
+++ sudoers (revision 2)
@@ -88,6 +88,6 @@
# Defaults targetpw # Ask for the password of the target user
# ALL ALL=(ALL) ALL # WARNING: only use this together with 'Defaults
targetpw'

-## Read drop-in files from /etc/sudoers.d
+## Read drop-in files from sudoers.d
## (the '#' here does not indicate a comment)
-##includedir /etc/sudoers.d
+# includedir sudoers.d
```

The output shows the `sudoers` file from line 88. The lines that were changed between version 1 and version 2 are marked with a preceding “+” or "-". A "-" denotes lines that were changed or deleted, and a "+" denotes updated or added lines.

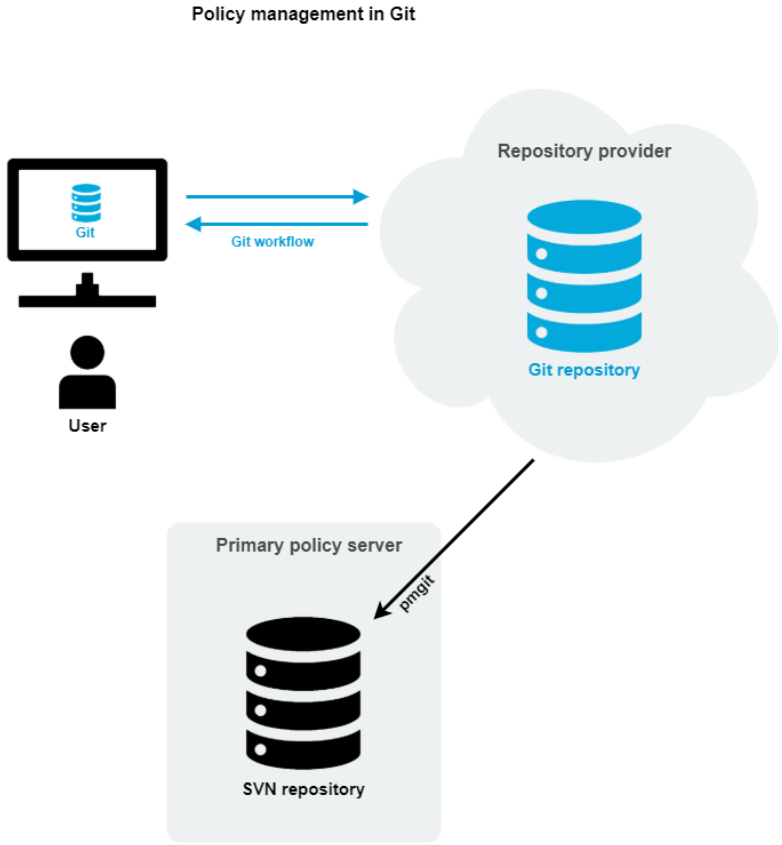
Managing policies in Git

The `pmgit` utility is a tool that can mediate version control operations between Subversion (SVN) and Git version control systems.

The `pmgit` utility uses the internal SVN policy repository to apply policies to the system, but it also uses an intermediate Git-SVN repository to keep the local SVN policy repository up-to-date from an external Git policy repository. You can manage this Git repository from outside the primary policy server.

The `pmgit` utility uses the `git` and `git svn` tools to perform the required version control operations, so you must install these tools on the Privilege Manager for Unix primary policy server. You can install both `git` and `git svn` using the system-specific package manager.

pmgit policy management concept



When you enable Git policy management, `pmgit` creates a backup from the original SVN policy repository which you can later restore if needed.

You can configure the Git policy management interactively by running the `pmgit --interactive` command.

There are two major advantages of the Git policy management:

- You can make changes in policies from anywhere, where the Git policy repository is available without the need to log into the policy server.

- You can use the "Git-workflow" by separating development and production branches, creating pull requests, having reviews before merge, and so on.

In this mode, you can no longer edit policies on the policy servers (neither primary, nor secondary). The `pmpolicy` command will reject every request that would make changes in the local SVN policy repository.

Under the hood, Privilege Manager for Unix policy servers still use the original SVN policy repository when updating policies, but the `pmgit` utility synchronizes the changes from Git to SVN using a local Git clone of the remote Git policy repository. The local copy is created at the following location:

```
/var/opt/quest/qpm4u/.qpm4u/.repository
```

Do not edit the local Git clone because it is maintained by the `pmgit` utility. Any changes made to the local Git clone will be discarded when `pmgit` synchronizes the changes from the remote Git policy repository.

The Git-SVN synchronization can either be manual or periodic, based on a predefined interval.

Before applying changes to the SVN policy repository, `pmgit` checks for syntax errors in the updated policy files. If no errors are found, `pmgit` creates a new SVN commit with the changes on top of the trunk. If a syntax check fails, `pmgit` logs the error reason via syslog. Optionally, `pmgit` can run a user-defined script (alert script) to take custom actions.

To enable Git policy management with the default settings, use the following command:

```
pmgit enable --export --git-url https://github.com/user/example.git
```

In this example, the Git policy repository at the specified URL must exist and must be an empty bare repository, or Git will reject the operation.

You can optionally configure the following settings:

- Git branch (Default: master)
- Update interval (Default: 5 minutes)
- Alert script (Default: N/A)

Each of the settings are stored in Privilege Manager's main configuration file (`pm.settings`).

For more information on `pmgit` subcommands, see pmgit.

Prerequisites for Git policy management

Before using Git policy management, do the following:

- Install the `git` tool on the primary policy server using the system-specific package manager.

- Install the `git svn` tool on the primary policy server using the system-specific package manager.
- Configure the `git` tool for passwordless authentication to allow automatic synchronization between the primary policy server and the remote Git repository.
- Enable Git policy management mode in the `pmgit` tool.

8

Administering Log and Keystroke Files

Safeguard allows you to control what is logged, as well as when and where it is logged. To help you set up and use these log files, the topics in this section explore enabling and disabling logging, as well as how to specify the log file locations.

Safeguard includes three different types of logging; the first two are helpful for audit purposes:

- **keystroke logging**, also referred to as I/O logging

  Keystroke logs record the user's keystrokes and the terminal output of any sessions granted by Safeguard.

- **event logging**

  Event logs record the details of all requests to run privileged commands. The details include what command was requested, who made the request, when the request was sent, what host the request was submitted from, and whether the request was accepted or rejected.

- **error logging**

You can configure some aspects of the event and keystroke logging by means of the security policy on the policy servers. What you can configure and how you configure it depends on which type of security policy you are using on your policy server -- pmpolicy or sudo.

Related Topics

Security policy types

Configuring keystroke logging for Safeguard for Sudo policy

Safeguard for Sudo enables event logging. Each time a sudo command is run, the policy server accepts or rejects the requested command according to the sudoers policy file and creates an event (audit) log. If enabled, the policy server records the keystroke input and terminal output for each *accepted* command, creating comprehensive "keystroke logs" files. With these logs, you can perform forensic-level auditing of any command that ran by means of sudo.

Event logs are captured and stored on the policy servers in `/var/opt/quest/qpm4u/pmevents.db`; keystroke logs are stored at `/var/opt/quest/qpm4u/iolog`.

You can use the `iolog_dir` and `iolog_file` policy options to reconfigure the iolog file location.

Configure the sudoers policy for keystroke logging by using the `log_input` and `log_output` defaults flags, or the `LOG_INPUT` and `LOG_OUTPUT` command tags, as follows:

```
Defaults log_input, log_output # keystroke logging enabled
Defaults!/sbin/reboot !log_input,!log_output # no logging for reboots
```

For complete I/O log records you must use both `log_input` and `log_output`.

```
# disable keystroke logging for the pmreplay command
ADMINS ALL = (ALL) NOLOG_INPUT:NOLOG_OUTPUT:/opt/quest/sbin/pmreplay
```

`ADMINS` is a `User_Alias`. See the *Sudoers man page* for definition of User_Alias.

Validating Sudo commands

To validate that the centrally managed policy is working, log on to a policy server (or a Sudo Plugin host) as a non-root user, run a command that is already set up in your sudoers policy file and observe the results.

Use a command you expect to work, such as:

```
$ sudo id
```

Then run a command that you know you do not have sufficient privileges to run. For instance, run a fake command, such as:

```
$ sudo fakecmd
```

When Safeguard rejects a command, it displays a message similar to this:

```
Sorry, user tuser is not allowed to execute ‘fakecmd’ as root on
myhost.example.com.
Request rejected by Safeguard
```

All systems that are joined to the same policy server will have the same results based on how you have the sudoers policy file configured.

Local logging

The location of the error logs for the Safeguard components, `pmrun` and `pmmasterd`, is specified using keywords in the `pm.settings` file. Enter the following to specify that you want the error logs written to the `/var/adm` directory:

```
pmmasterdlog /var/adm/pmmasterd.log
pmrunlog /var/adm/pmrun.log
```

Alternatively, you can enable UNIX syslog error logging in the `pm.settings` file, by specifying:

```
syslog YES
```

Use one of the following keywords to specify which syslog facility to use:

- LOG_KERN
- LOG_USER
- LOG_MAIL
- LOG_DAEMON
- LOG_AUTH (the default)
- LOG_LPR
- LOG_NEWS
- LOG_UUCP
- LOG_CRON
- LOG_LOCAL0 through LOG_LOCAL7

For example, to enable syslog error logging using the LOG_AUTH facility, enter in the `pm.settings` file:

```
syslog YES
facility LOG_AUTH
```

See PM settings variables on page 112 for more information about modifying the Safeguard configuration settings.

Event logging

Event logs are enabled by default for all requests sent to the Safeguard Policy Servers. The default location of the event log file is `/var/opt/quest/qpm4u/pmevents.db`.

Keystroke (I/O) logging

Once your 30-day trial license has expired, One Identity requests that you obtain a Keystroke Logging license to remain in compliance. See Safeguard licensing on page 10 for details.

You can enable keystroke logging using the `log_input` and `log_output` default parameters.

Enabling `log_input` and `log_output` enables keystroke logging.

For example, to enable keystroke logging for all requests, specify:

```
Defaults log_input, log_output
```

To specify keystroke logging of output just for the `root` user, specify:

```
Defaults:root log_output
```

You can also override default settings by using the `LOG_INPUT`, `LOG_OUTPUT`, `NOLOG_INPUT`, `NOLOG_OUTPUT` tags in a user specification entry. For example, to suppress keystroke logging for the `ls` command, enter:

```
ALL ALL=(ALL) NOLOG_OUTPUT:/bin/ls
```

The location of the keystroke log file is determined by the `iolog_dir` and `iolog_file` default specifications.

The defaults are:

```
Defaults iolog_dir = "/var/opt/quest/qpm4u/iolog"
Defaults iolog_file = "%{user}/%{runas_user}/%{command}_%Y%m%d_%H%M_XXXXXX"
```

See the Sudoers man page for an explanation of the supported percent (%) escape sequences.

The trailing "XXXXXX" characters at the end of `iolog_file` are required; without them, no I/O log will be generated. These X's are replaced with a unique combination of digits and letters, similar to the `mktemp()` function.

Audit server logging

Administrators can stream event logs and keystroke (IO) logs from a client to a sudo log audit server (or compatible server) that implements the sudo logsrv protocol. This feature is disabled by default. Enable the recording service through configuring the policy server with `pmsrvconfig` or by editing `pm.settings`.

The stored keystroke (IO) logs can be rotated, trimmed, and compressed to manage storage space.

A syslog output of streamed keystroke (IO) logs can be used to send the data to a Security Information and Event Management (SIEM) tool.

`pmmasterd` sends I/O logs to the audit server when a command is run via pmrun. I/O logs are sent in real-time. A setting in `pm.settings` determines whether I/O logs are stored locally too.

Configuration options

You can configure the audit server in `pm.settings` or interactive mode The `pm.settings` file sincludes settings for the CA bundle, client certificate, and client key files as well as other settings.

Configuration with `pm.settings`

One or more audit servers can be specified in the `pm.settings` file along with the associated port (which defaults to port 30344).

When `pmmasterd` receives an event from the client, it relays the event to `sudo_logsrvd`. Events that are supported include: Accept, Reject, and Alert. Logging to the audit server is in addition to local logging. A setting in the `pm.settings` file specifies whether an unreachable audit server is considered a fatal error or not.

See PM settings variables for more information about modifying the following configuration settings:

- auditsrvCAbundle
- auditsrvCert
- auditsrvEnabled
- auditsrvEnforced
- auditsrvHosts
- auditsrvKeepalive
- auditsrvLocaliologs
- auditsrvLogdir
- auditsrvPkey

- auditsrvPSpaceMB
- auditsrvTimeout
- auditsrvTLS
- auditsrvTLSCheckpeer
- auditsrvTLSVerify

Configuration with `pmsrvconfig`

You can also use the interactive mode of `pmsrvconfig` to perform most configuration.

Example for interactive mode

In this example, you can see the how interactive mode works.

```
$ pmsrvconfig -i
[...]
** Where would you like to store errors reported by the Privilege Manager
policy server daemon? [/var/log/pmmasterd.log]
- Policy server log location: /var/log/pmmasterd.log
*** Configure Audit Server for Privilege Manager
** Audit Server configuration for pmmasterd
- The Audit Server can receive event and kestroke logs in real time.
- If enabled, pmmasterd streams all logs to the Audit Server.
** Would like you to configure Audit Server(s) for Privilege Manager [YES]
- Configuring Audit Server(s) for pmmasterd: YES
** Audit Server availability
- If none of the configured audit servers are available, the policy server
can either
- - Reject all commands until an audit server becomes available
- - Save audit trails locally on the policy server.
These trails will be transferred automatically to an audit server when it
becomes available.
- When configured audit server(s) become unavailable,
- 1) I want the policy server to reject all requests
- 2) I want to use audit trail caching on the policy server
** Please select an option [1] 2
** Enter the directory where pmmasterd can save audit trails
[/var/opt/quest/qpm4u/auditserver]
- Audit trails will be saved to directory:
/var/opt/quest/qpm4u/auditserver
** How much disk space shall be preserved in megabytes? [100]
- Command execution will not be permitted if the available disk space
drops below
100 megabytes
```

```
** Would you like to retain old format IO logs locally? [YES]
- Retaining old IO logs locally: YES
** Enter connection timeout in seconds: [3] 10
- Connection timeout: 10
** Would you like to enable TCP keepalive messages? [YES]
- TCP keepalive messages enabled: YES
** Would you like to secure connection with TLS? [YES]
- Communication between policy server and audit server is secured with
TLS: YES
** Audit Servers are already configured:
- qpmdevel1.qpmdomain:30344
** Would you like to reconfigure the Audit Servers? [NO]
- Overwriting Audit Server list: YES
** Please enter the address (hostname | ip_v4 | ip_v6): 127.0.0.1
- Audit Server address: 127.0.0.1
** What port number would you like to use for the audit server daemon?
[30344]
- Audit Server port: 30344
** Do you want to add an additional Audit Server to the configuration?
[NO]
- 127.0.0.1:30344** Configure TLS parameters
- You need to provide the following files in order to configure TLS:
- * CA bundle file
- * Private key file
- * Certificate file
** Please enter the full path to the CA bundle file
[/etc/ssl/sudo/ca.bundle.pem]:
** Checking that CA bundle is in PEM format [ OK ]
- CA bundle file is set: /etc/ssl/sudo/ca.bundle.pem
** Please enter the full path to the private key file
[/etc/ssl/sudo/qpm_qpmdevel1.key.pem]:
** Checking that private key is in PEM format [ OK ]
- Private key file is set: /etc/ssl/sudo/qpm_qpmdevel1.key.pem
** Please enter the full path to the certificate file
[/etc/ssl/sudo/qpm_qpmdevel1.cert.pem]:
** Checking certificate against the private key [ OK ]
** Checking certificate chain of trust [ OK ]
** Checking certificate expiration [ OK ]
** Checking hostname/IP address [WARN]
- WARNING: Could not verify hostname/IP
- Client certificate file is set:
/etc/ssl/sudo/qpm_qpmdevel1.cert.pem
** Would like you to check connection to the audit server(s)? [YES]
```

Using `pmsrvconfig`

You can use the `pmauditsrv` and options for the following:

- Verifies that the configured audit servers are accessible and configured properly and exchanges a "hello" message with the server.
- If the audit server is not accessible, stores the events and keystroke (IO) logs temporarily offline and sent to the audit server when it is available.

The connection from `pmmasterd` to `sudo_logsrvd` uses TLS to secure data transmission. If none of the audit servers are reachable, event logs and keystroke I/O logs are queued locally on the policy server and sent to the audit server once it is available. Offline logs are encrypted until they are transferred to the log server.

For more information, see pmauditsrv.

Viewing the log files using command line tools

Using command line tools, you can list events and replay log files directly from the primary policy server using the `pmlogsearch`, `pmreplay`, and `pmremlog` commands.

`pmlogsearch`

`pmlogsearch` is a simple search utility based on common criteria. Run `pmlogsearch` on the primary server to query the logs on all servers in the policy group. `pmlogsearch` provides a summary report on events and keystroke logs matching at least one criteria. `pmlog` provides a more detailed report on events than `pmlogsearch`.

Hostnames may appear in the event logs and keystroke log files in either fully qualified format (`myhost.mycompany.com`) or in short name format (`myhost`), depending on how hostnames are resolved and the use of the short name setting in the `pm.settings` file. To ensure that either format is matched, use the short host name format with an asterisk wildcard (`myhost*`) when specifying a hostname search criteria.

See pmlogsearch on page 156 for more information about the syntax and usage of the `pmlogsearch` command.

`pmlogsearch` performs a search across all policy servers in the policy group and returns a list of events (and associated keystroke log file names) for requests matching the specified criteria. You specify search criteria using the following options (you must specify at least one search option):

Table 9: Search criteria options

Command	Description
--after "YYYY/MM/DD hh:mm:ss"	Search for sessions initiated after the specified date and time.
--before "YYYY/MM/DD hh:mm:ss"	Search for sessions initiated before the specified date and time.

Command	Description
--host hostname	Search for sessions that run on the specified host.
--result accept|reject	Return only events with the indicated result.
--text keyword	Search for sessions containing the specified text.
--user username	Search for sessions by the specified requesting user.

The following `pmlogsearch` options support the use of wildcards, such as ***** and **?**:

- --host
- --user

To match one or more characters, you can use wild card characters (such as ? and *) with the `--host`, `--text`, and `--user` options; but you must enclose arguments with wild cards in quotes to prevent the shell from interpreting the wild cards.

If there is a keystroke log associated with the event, it displays the log host and pathname along with the rest of the event information.

The following example lists two events with keystroke (IO) logs:

```
      # pmlogsearch --user sally
Search matches 2 events
2013/03/16 10:40:02 : Accept : sally@qpmsrv1.example.com
   Request: sally@qpmsrv1.example.com : id
   Executed: root@qpmsrv1.example.com : id
   IO Log: qpmsrv1.example.com:/opt/quest/qpm4u/iologs/demo/sally/id_20120316_
1040_ESpL6L
2013/03/16 09:56:22 : Accept : sally@qpmsrv2.example.com
   Request: sally@qpmsrv2.example.com : id
   Executed: root@qpmsrv2.example.com : id
   IO Log: qpmsrv2.example.com:/opt/quest/qpm4u/iologs/demo/sally/id_20120316_
0956_mrVu4I
```

pmreplay

You can use the `pmreplay` command to replay a keystroke log file if it resides on the local policy server.

To replay the log, run:

```
# pmreplay <path_to_keystroke_log>
```

For example, the following command replays the first `ls -l /etc` log from the previous example:

```
# pmreplay /opt/quest/qpm4u/iologs/demo/sally/id_20120316_1040_ESpL6L
```

pmremlog

If the keystroke log resides on a remote policy server, you can use the `pmremlog` command with the `-h <remote_host>` and `-p pmreplay` options to remotely replay a keystroke log file. You specify the path argument to the remote `pmreplay` after the `--` flag.

For example, enter the following command all on one line:

```
# pmremlog -h qpmsrv2 -p pmreplay -- /opt/quest/qpm4u/iologs/demo/sally/id_
20120316_0956_mrVu4I
```

Host names may appear in the event logs and keystroke log files in either fully qualified format (`myhost.mycompany.com`) or in short-name format (`myhost`), depending on how host names are resolved and the use of the `shortnames` setting in the `pm.settings` file. To ensure that either format is matched, when you specify a host name search criteria, use the short-host name format with an asterisk wild card (For example, `myhost*`).

Listing event logs

You can list the events that are logged when you run a command, whether accepted or rejected by the policy server.

Keystroke logs are related to events. When you run a command, such as `sudo whoami`, the policy server either accepts or rejects the command based on the policy. When the policy server accepts the command, it creates an event and a corresponding keystroke log. If it rejects the event, it does not create a keystroke log. In order to view a keystroke log, you must first list events to find a particular keystroke log.

The `pmlog` command displays event log entries, such as events by date and time, host, user, run user, command, and result.

To display a list of events from the command line on the policy server

1. From the command line, enter:

   ```
   # pmlog --after "2011/05/06 00:00:00" --user "tuser"
   ```

   `pmlog` provides direct and flexible access to the event logs on the local policy server and is capable of complex queries.

   If you run a command, you might see output similar to the following which indicates the policy server has successfully accepted or rejected commands:

   ```
   Accept 2011/05/11 13:20:04 tuser@ myhost.example.com -> root@
   myhost.example.com
      whoami
      Command finished with exit status 0
   Accept 2011/05/11 14:05:58 tuser@ myhost.example.com -> root@
   ```

```
myhost.example.com
   whoami
   Command finished with exit status 0
Reject 2011/05/11 14:06:17 tuser@ myhost.example.com
   Fakecmd
```

The following `pmlog` options support the use of wildcards, such as ***** and **?**:

- --user
- --runuser
- --reqhost
- --runhost
- --masterhost

You can also use the `pmremlog` command on the primary policy server to run `pmlog` on secondary policy servers. For example:

```
# pmremlog -h polsrv2 -p pmlog -- --user myuser --command sh
```

Related Topics

pmlog

pmremlog

Backing up and archiving event and keystroke logs

Use the `pmlogadm` program to perform backup or archive operations on a policy server's event log database. Because Safeguard stores keystroke logs in individual flat files on the policy server, you may use standard Unix commands to back up or archive them. Make sure the keystroke log files are not associated with active sessions prior to backup or archive.

Disabling and enabling services

While `pmlogadm` can perform the backup and archive operations on a live event log database, for best results we recommend that you follow these steps prior to performing a backup or archive.

1. Stop the `pmserviced` and `pmlogsrvd` services.
   This example shows how to disable services on Redhat Linux systems:

```
# service pmserviced stop
Stopping pmserviced service:     done
# service pmlogsrvd stop
Stopping pmlogsrvd service:     done
```

2. Ensure there are no running `pmmasterd` processes:

   ```
   # ps -ef | grep pmmasterd
   ```

   A running `pmmasterd` process indicates that there may be an active Safeguard session.

This procedure also allows you to safely backup or archive any keystroke log files. Once the backup or archive operation has completed, remember to restart the `pmserviced` and `pmlogsrvd` services.

This example shows how to restart the services on Redhat Linux systems:

```
# service pmlogsrvd start
Starting pmlogsrvd service:     done
# service pmserviced start
Starting pmserviced service:     done
```

Backing up event logs

The `pmlogadm` backup command creates a clean backup copy of your event log database.

This example performs a backup of the current event log database, placing the copy in the `/backup` directory:

```
# pmlogadm backup /var/opt/quest/qpm4u/pmevents.db /backup
5 / 208 pages complete
10 / 208 pages complete
...
205 / 208 pages complete
208 / 208 pages complete
```

Backing up keystroke logs

Safeguard stores the keystroke logs in individual files and do not require any special commands for processing.

This example uses the `unix cp` command to recursively copy the keystroke logs to the `/backup` directory:

```
# cp -r /var/opt/quest/qpm4u/iolog /backup
```

Archiving event logs

The `pmlogadm` archive command creates an archive of old event logs and removes the old event logs from the current database. The following example archives logs for all events that occurred before April 1, 2014 from the current event log database, creating an archive database in the `/archive/2014Q1` directory.

If you omit the `--no-zip` option, `pmlogadm` also creates a tar-gzip'ed archive of the database files.

```
# pmlogadm archive /var/opt/quest/qpm4u/pmevents.db 2014Q1 \
  --dest-dir /archive --no-zip --before "2014-04-01 00:00:00"
Archive Job Summary
     Source Log : /var/opt/quest/qpm4u/pmevents.db
   Archive Name : 2014Q1
Destination Dir : /archive
    Zip Archive : No
   Cut off time : 2014/04/01 00:00:00

No pmlogsrvd pid file found, assuming service is not running.
X events will be archived.
Adding events to the archive.
Verifying archive.
Archive verification completed successfully. Removing events from source log.
Archive task complete.
```

Archiving keystroke logs

You can use the `pmlog` command with some carefully chosen options to get a list of keystroke logs associated with the event logs you archive. In this example, you process the list generated by `pmlog`, with the Unix `xargs` and `mv` commands to move the keystroke logs into the `/archive/2014Q1/iolog` directory.

```
# mkdir /archive/2014Q1/iolog
# pmlog -f /archive/2014Q1/archive.db \
   -c "defined iolog && length(iolog) != 0" -p iolog \
   | xargs -i{} mv {} /archive/2014Q1/iolog
```

The usage of the `xargs` command may differ depending on your platform.

Supported sudo plugins

Safeguard for Sudo supports loading the following sudo-compatible plugins on the policy server:

- Approval plugin
- Audit plugin

You can write these sudo plugins both in C and in Python.

To load a sudo audit or approval plugin on a policy server, you must configure the plugins in the `sudoers` policy file located in `/etc/opt/quest/qpm4u/policy/sudoers` by default. On a policy server that supports multiple policies, you can have different plugins configured for each policy.

Syntax

Safeguard for Sudo does not use the `/etc/sudo.conf` file to load modules. Safeguard for Sudo uses the `sudoers` policy file and it uses a slightly different syntax.

The syntax of the `/etc/sudo.conf` file is the following:

```
Plugin symbol_name plugin_file.so plugin_arguments...
```

The syntax of the Safeguard for Sudo `sudoers` policy file is the following:

```
Defaults plugins += "symbol_name plugin_file.so plugin_arguments..."
```

Where:

- `symbol_name` is the name of the symbol used to look up the plugin.
- `plugin_file.so` is the path to the plugin file.
- `plugin_arguments` are optional arguments passed to the plugin. For Python plugins, the arguments are used to find the Python script to load.

For more information about the audit and approval plugins, see the *Sudo Plugin API* and *Sudo Python Plugin API* man pages.

Configuring a sudo approval plugin

Sudo version 1.9 introduced a new plugin API to apply extra restrictions to a command after it has been accepted by the sudoers policy. Safeguard for Sudo supports loading sudo-compatible approval plugins, including those written in Python, on the policy server. You can specify multiple approval plugins in the `sudoers` file. Safeguard for Sudo currently supports loading up to 8 Python approval plugins at once.

For more information about configuring a C-based approval plugin, see the *Sudo Plugin API man page*.

Prerequisites

- Install Sudo version 1.9 or newer.
- To use plugins written in Python:
  - Install Python version 3.0 or newer.
  - Install the sudo-python package, available at https://www.sudo.ws/download.html.

To configure a Python-based approval plugin in the `sudoers` file

To configure the sudoers policy to load the Python-based approval plugin, use the following configuration in the `sudoers` file:

```
Defaults plugins += "python_approval python_plugin.so ModulePath=<path>
ClassName=<class>"
```

Where `ModulePath` is the path to the Python script that the plugin uses, and `ClassName` denotes what gets called within the plugin.

The following example Python approval plugin only allows users running commands during business hours, that is, from Monday to Friday between 8:00 and 17:59:59.

```
Defaults plugins += "python_approval python_plugin.so \
                       ModulePath=/root/example_approval_plugin.py \
                       ClassName=BusinessHoursApprovalPlugin"
```

```
Defaults plugins += "python_approval python_plugin.so \
ModulePath=/root/example_approval_plugin.py \
ClassName=BusinessHoursApprovalPlugin"
```

For a more detailed Python approval plugin example, see the sudo repository on GitHub.

Configuring a sudo audit plugin

Sudo version 1.9 introduced a new plugin API to access audit information. Safeguard for Sudo supports loading sudo-compatible audit plugins, including those written in Python, on the policy server. This can be used in a number of different ways, for example to implement

custom logging or to send events from Safeguard for Sudo directly to Elasticsearch or other Logging as a Service providers.

You can specify multiple audit plugins in the `sudoers` file. Sudo currently supports loading 8 Python audit plugins at once.

For more information about configuring a C-based audit plugin, see the *Sudo Plugin API man page*.

Prerequisites

- Install Sudo version 1.9 or newer.
- To use plugins written in Python:
  - Install Python version 3.0 or newer.
  - Install the sudo-python package, available at https://www.sudo.ws/download.html.

To configure a Python-based audit plugin in the `sudoers` file

To configure the `sudoers` policy to load the Python-based audit plugin, use the following configuration in the `sudoers` file:

```
Defaults plugins += "python_audit python_plugin.so ModulePath=<path>
ClassName=<class>"
```

The following example Python audit plugin logs the plugin accept / reject / error results to the output:

```
Defaults plugins += "python_audit python_plugin.so \
                     ModulePath=/root/example_audit_plugin.py \
                     ClassName=SudoAuditPlugin"
```

For a more detailed Python audit plugin example, see the sudo repository on GitHub.

9

Troubleshooting

To help you troubleshoot, One Identity recommends the following resolutions to some of the common problems you might encounter as you deploy and use Safeguard.

Enabling sudo policy debug logging

Debug logs can help you determine if the sudo options are being enabled correctly in the policy.

To enable debug logging for Sudo policy

1. Add a debug line to the `/etc/sudo.conf` file. For example, to log debug and trace information to the file `/var/log/sudo_debug`, add:

   ```
   Debug sudo /var/log/sudo_debug all@debug
   ```

For systems without a `/var/log` directory, use `/var/adm/sudo_debug` instead.

Enabling tracing for Sudo Plugin

Since the Sudo Plugin is not a program, the `/tmp/pmplugin.ini` file needs be manually created in order to enable tracing for the Sudo Plugin itself.

To create the .ini file to enable tracing for the Sudo Plugin

1. Run the following as root:

   ```
   printf 'FileName=/tmp/pmplugin.trc\nLevel=0xffffffff\n' >
   /tmp/pmplugin.ini
   ```

2. Once you have finished getting the trace output you need, remove the `/tmp/pmplugin.ini` file to disable tracing.

Join fails to generate a SSH key for sudo policy

If you attempt to join a Sudo Plugin host and see a ssh-keyscan failure message similar to this:

```
** Generate ssh key [FAIL]
    - failed to update known_hosts file:getaddrinfo <myhost>: Name or
service not known
```

You might be using an unresolvable, short host name (as `myhost` in the above example) instead of the fully qualified domain name.

To workaround this issue, add the domain to the search line in the `/etc/resolv.conf` file.

Join to policy group failed on Sudo Plugin

When you join a host with the Sudo Plugin to a policy group you are required to enter a password. The *Join* password is the password for the `pmpolicy` user that was set when the `qpm-server` was configured. See Configuring the Safeguard for Sudo Primary Policy Server on page 19 for more information about `pmpolicy` service account.

If the `Join` operation does not recognize the `pmpolicy` user password, you will receive an error message with the following snippet:

```
Enter join password for remote user:pmpolicy@example.com:

[FAIL]
   - Failed to copy file using ssh.
   - Error: Failed to add the host to the list of known hosts
      (/var/opt/quest/qpm4u/pmpolicy/.ssh/known_hosts).
      Permission denied (gssapi-keyex,gssapi-with-mic,publickey,keyboard-
interactive).

   ** Failed to setup the required ssh access.
   ** The pmpolicy password is required to copy a file to the primary
   ** policy server.
   ** To complete this configuration, please rerun this command and
   ** provide the correct password.
```

```
      - ERROR: Failed to configure pmclient user
      - ERROR: Configuration of qpm4u unsuccessful.
      - ERROR: Installation log file is
        /opt/quest/qpm4u/install/pmjoin_plugin_output_20121022.log
[1][root@sles10-qa ~]#
```

Run the `Join` operation again entering a correct password.

Load balancing and policy updates

`pmpluginloadcheck` is both a command and a background daemon (run with the `-i` flag). When run as a command, it checks, updates, and reports on the status of the policy server. You can use `pmpluginloadcheck` from a Sudo Plugin host.

When run as a daemon process, it keeps track of the status of the policy servers for failover and load-balancing purposes. On policy servers, `pmpluginloadcheck` is responsible for keeping the production policy file up to date for the offline policy cache.

See pmpluginloadcheck on page 164 for more information about the syntax and usage of this command.

Policy servers are failing

The primary and secondary policy servers must be able to communicate with each other and the remote hosts must be able to communicate with the policy servers in the policy group.

For example, if you run `pmpluginloadcheck` on a Sudo Plugin host to determine that it can communicate with other policy servers in the group, you might get output similar to the following:

```
++ Checking host:myhost.example.com (10.10.181.87) ... [FAIL]
```

There are several possible reasons for failure:

- Policy server host is down
- Network outage
- Service not running on policy server host

pmgit Troubleshooting

This section describes common issues that may occur when using `pmgit`. Follow the instructions to troubleshoot `pmgit` operation.

Setting alert for syntactically incorrect policies

Since policy edits are not locally bound to the policy server when using Git policy management, syntactically incorrect policies can enter the Git repository. To address such cases, set an alert from the policy server to warn you if the policy is incorrect.

As an administrator, you can use your own alert script which `pmgit` tool can call if the policy syntax checking returns an error message after the synchronization between the Git policy repository and the SVN policy repository.

If an alert script is configured, the `pmgit` tool calls it with 2 parameters:

- Email address from the last Git commit
- Error message from the syntax check

Sample script

This is a sample script in bash which sends the error message to the user who initiated the last commit.

```
#!/bin/bash

email_address="$1"
shift
error_msg="$@"

/usr/sbin/sendmail -F "noreply" "${email_address}" <<EOF
subject:pmgit error

Syntax error occured in one of the policy files:
"${error_msg}"
EOF
```

To set pmgit tool to send alert messages based on your alert script, see pmgit Set.

Automatic synchronization failed

Error

After a successful Git policy management configuration and automatic update interval setting, Syslog sends the error message:

```
pmgit: Failed to fetch <Git:_URL>.: Permission denied, please try again.
<user>@<host>: Permission denied (publickey,password)
```

Cause

You have not configured Git for passwordless authentication.

Effect

Automatic synchronization between Git and SVN is not working because `pmgit update` cannot run in the background due to a password prompt.

Solution

Configure Git to allow Git operations from the policy server towards the remote repository.

Failed to push references to Git URL

Error

After export pmgit sends the error message:

```
# pmgit export --git-url <Git_URL>
Creating backup from SVN repository ...                              [ OK ]
Creating directory for local Git repository ...                      [ OK ]
Cloning SVN ...                                                      [ OK ]
Setting Git remote ...                                               [ OK ]
Push Git repository to remote ...                                  [ ERROR ]
   To <Git_URL>
! [rejected]        <Git_branch> -> <Git_branch> (fetch first)
error: failed to push some refs to '<Git_URL>'
```

Cause

You tried to export to a Git repository which is not empty.

Effect

You are unable to export the policies to that Git repository.

Solution

Create an empty bare repository.

Example

This is an example for creating an empty bare Git repository from command line.

```
git init --bare <repo_name>.git
```

Sudo command is rejected by Safeguard for Sudo

Safeguard for Sudo might reject a sudo command. For example, let us assume you ran the following command:

```
$ sudo id
```

and received output similar to the following:

```
<user> is not in the sudoers file. This incident will be reported.
Request rejected by Safeguard
```

There are several things you can do to troubleshoot this issue.

To troubleshoot why a sudo command is rejected

Run the following from the policy server:

1. To ensure the user has permission, run the following as a sudo administrator.

   ```
   # sudo -U <username> -l
   ```

2. To check that the policy is located at `/etc/opt/quest/qpm4u/policy/sudoers` is the current version, run:

   ```
   # pmpolicy masterstatus
   ```

In the output, ensure that *Current Revision* and *Latest Trunk Revision* have the same number and *Locally modified* is "No".

3. To ensure the user has permission to run the command, check the `/etc/opt/quest/qpm4u/policy/sudoers` file and verify the user's (or group's) permissions:

```
# cat /etc/opt/quest/qpm4u/policy/sudoers
```

4. To verify that the policy server is working properly, enter:

```
# pmsrvcheck
```

This command returns output similar to:

```
testing policy server [ Pass ]
```

From the command line, enter:

```
# pmsrvinfo
```

This command returns output similar to:

```
Policy Server Configuration:
----------------------------
   Safeguard version :
   7.2
  .0 (0nn)
   Listening port for pmmasterd daemon  : 12345
   Comms failover method                : random
   Comms timeout(in seconds)            : 10
   Policy type in use                   : sudo
   Group ownership of logs              : pmlog
   Group ownership of policy repository : pmpolicy
   Policy server type                   : primary
   Primary policy server for this group : Myhost1
   Group name for this group            : Myhost1.example.com
   Location of the repository           : file:
                        ////var/opt/quest/qpm4u/.qpm4u/.repository/sudo_
repos/trunk
   Hosts in the group : Myhost1
```

Related Topics

pmpolicy

pmsrvcheck

pmsrvinfo

Sudo policy is not working properly

If your sudo policy is not working as expected, use these troubleshooting steps:

1. To verify the version of sudo on your host:

   ```
   # sudo -V
   ```

2. To verify that the Sudo Plugin host is joined to the policy server, run:

   ```
   # pmpluginfo
   ```

3. To see what commands the user is allowed to run:

   ```
   # sudo -l -U <username>
   ```

   This command returns output similar to:

   ```
   Matching Defaults entries for testuser on this host:
         log_output
   User testuser may run the following commands on this host:
         (ALL) /opt/quest/bin/
   ```

4. On the policy server, use the `pmpolicy` utility for managing the Privilege Manager for Unix security policy.
   a. To verify that you have the correct version of the policy, run:

      ```
      # pmpolicy masterstatus
      ```

      Ensure that **Locally modified** in the output is **No**.

   b. To update the version of the policy, run:

      ```
      # pmpolicy sync
      ```

   c. To verify there are no syntax errors in the policy, run:

      ```
      # pmpolicy checkout -d <dir>
      ```

5. On the Sudo Plugin host, use the `pmpolicyplugin` utility to display the revision status of the cached security policy on this host or to request an update from the central repository.
   a. To verify that you have the correct version of the policy on the Sudo Plugin host, run

      ```
      # pmpolicyplugin
      ```

Use the `-g` option to update the local cached security policy with the latest revision on the central repository (equivalent to `pmpolicy sync` on a server).

Related Topics

pmplugininfo

pmpolicy

pmpolicyplugin

Appendix A

Safeguard Variables

This appendix provides detailed information about the variables that may be present in event log entries:

- Global input variables
- Global output variables
- Global event log variables
- PM settings variables

See also Profile Variables for additional information about policy profile variables.

Global input variables

The following predefined global variables are initialized from the submit-user's environment.

Table 10: Global input variables

Variable	Data type	Description
alertkeymatch	sting	The pattern matched by `pmlocald`.
argc	integer	Number of arguments in the request.
argv	list	List of arguments in the request.
client_parent_pid	integer	Process ID of the client's parent process.
client_parent_uid	integer	User ID associated with the client's parent process.
client_parent_procname	string	Process name of a client's parent process.
clienthost	string	Originating login host.

Variable	Data type	Description
command	string	Pathname of the request.
cwd	string	Current working directory.
date	string	Current date.
day	integer	Current day of month as integer.
dayname	string	Current day of the week.
domainname	string	The Active Directory domain name for the submit user if Authentication Services is configured.
env	list	List of submit user's environment variables.
false	integer	Constant value.
FEATURE_LDAP	integer	Read-only constant used with `feature_enabled()` function.
FEATURE_VAS	integer	Read-only constant used with `feature_enabled()` function.
gid	integer	Group ID of the submitting user's primary group on sudo host.
group	string	Submit user's primary group.
groups	list	Submit user's secondary groups.
host	string	Host destined to run the request.
hour	integer	Current hour.
masterhost	sting	Host on which the master process is running.
masterversion	string	Safeguard version of `masterhost`.
minute	integer	Current minute.
month	integer	Current month.
nice	integer	`nice` value of the submit user's login.
nodename	string	Hostname of the sudo client.
optarg	integer	Contains the parameter for the last argument or empty string.
opterr	integer	Determines whether to display errors from the `getopt` functions.
optind	integer	Contains the current argument list index. Use with `getopt` functions.
optopt	string	Contains the letter of the last option that had an

Variable	Data type	Description
		issue. Use with `getopt` functions.
optreset	boolean	Restarts the `getopt` functions from the beginning.
optstrictparameters	boolean	Lets `getopt_long()` recognize non-compliant argument parameter forms.
pid	integer	Process ID of the master process.
pmclient_type	integer	The type of client that sent the request.
pmclient_type_pmrun	integer	Read-only constant for pmrun type clients.
pmclient_type_sudo	integer	Read-only constant for sudo type clients.
pmshell	integer	Identifies a Privilege Manager for Unix shell program.
pmshell_builtin	integer	A constant value that identifies a shell `builtin` command.
pmshell_cmd	integer	Identifies a command run from a Privilege Manager for Unix shell program.
pmshell_cmdtype	integer	Identifies type of a shell subcommand.
pmshell_exe	integer	A constant value that identifies a normal executable command.
pmshell_interpreter	integer	Identifies the program directive of a shell script.
pmshell_prog	string	Name of the Privilege Manager for Unix shell program.
pmshell_script	integer	A constant value that identifies a shell script.
pmshell_uniqueid	string	`uniqueid` of the Privilege Manager for Unix shell program.
pmversion	string	SafeguardPrivilege Manager for Unix version string of client.
ptyflags	string	Identifies `ptyflags` of the request.
requestlocal	integer	Indicates if the request is local.
requestuser	string	User that the submit user wants to run the request.
rlimit_as	string	Controls the maximum memory that is available to a process.
rlimit_core	string	Controls the maximum size of a core file.

Variable	Data type	Description
rlimit_cpu	string	Controls the maximum size CPU time of a process.
rlimit_data	string	Controls the maximum size of data segment of a process.
rlimit_fsize	string	Controls the maximum size of a file.
rlimit_locks	string	Control the maximum number of file locks for a process.
rlimit_memlock	string	Controls the maximum number of bytes of virtual memory that can be locked.
rlimit_nofile	string	Controls the maximum number of files a user may have open at a given time.
rlimit_nproc	string	Controls the maximum number of processes a user may run at a given time.
rlimit_rss	string	Controls the maximum size of the resident set (number of virtual pages resident at a given time) of a process.
rlimit_stack	string	Controls the maximum size of the process stack.
samaccount	string	The sAMAccountName for the submit user if Authentication Services is configured.
selinux	integer	Identifies whether a client is running an SELinux environment.
status	integer	Exit status of the most recent system command.
submithost	string	Name of the submit host.
submithostip	string	IP address of the submit host.
thishost	string	The value of the `thishost` setting in `pm.settings` on the client.
time	string	Current time of request.
true	integer	Read-only constant with a value of 1.
ttyname	string	`ttyname` of the submit request.
tzname	string	Name of the time zone on the server at the time the event was read from the event log by pmlog.
uid	integer	User ID of the submitting user on host.
umask	integer	`umask` of the submit user.

Variable	Data type	Description
unameclient	list	`Uname` output on host.
unamemaster	list	`Uname`output on policy server host.
uniqueid	string	Uniquely identifies a request in the event log.
use_rundir	string	Contains the value "!~!" and represents the `runuser`'s home directory on the `runhost`.
use_rungroup	string	Contains the value "!g!" and represents the `runuser`'s primary group on the `runhost`.
use_rungroups	string	Contains the value "!G!" and represents the `runuser`'s secondary group list on the `runhost`.
use_runshell	string	Contains the value "!!!" and represents the `runuser`'s login shell on the `runhost`.
user	string	Submit user.
year	integer	Year of the request (YY).

argc

Description

Type **integer** READONLY

`argc` contains the number of arguments supplied for the original command. This includes the command name itself. For example, if the original command is `sudo ls -al`, then `argc` is set to 2.

Related Topics

argv

argv

Description

Type **list** READONLY

`argv` is a list of the arguments supplied for the original command, including the command itself. For example, if the original command is `sudo ls -al`, then `argv` is set to `{"ls","-al"}`.

Related Topics

argc

client_parent_pid

Description

Type **integer** READONLY

Process ID of client's parent process.

Related Topics

client_parent_uid

client_parent_procname

client_parent_uid

Description

Type **integer** READONLY

User ID associated with the client's parent process.

Related Topics

client_parent_pid

client_parent_procname

client_parent_procname

Description

Type **string** READONLY

Process name of a client's parent process.

Related Topics

client_parent_pid

client_parent_uid

clienthost

Description

Type **string** READONLY

`clienthost` contains the host name/IP address of the requesting host. For a Safeguard for Sudo command, this will be identical to the `submithost` variable. Always use short names when checking the `clienthost` variable, as some login programs may truncate the full host name.

Related Topics

submithost

submithostip

command

Description

Type **string** READONLY

The name of the command being run.

The `command` variable generally contains the full path name of the command being run. Use the `basename()` function to get the command name without the full path.

cwd

Description

Type **string** READONLY

`cwd` contains the pathname of the submit user's current working directory.

date

Description

Type **string** READONLY

`date` contains the date the request was submitted in the form: YYYY/MM/DD.

Related Topics

dayname
minute
hour
day
month
year
time

day

Description

Type **integer** READONLY

day contains the day the request was submitted formatted as an integer in the range: 1–31.

Related Topics

dayname
minute
hour
date
month
year
time

dayname

Description

Type **string** READONLY

dayname contains the abbreviated name ("Mon", "Tue, "Wed", "Thu", "Fri", "Sat" or "Sun") of the day the request was submitted.

Related Topics

minute
hour

day

month

year

time

date

domainname

Description

Type **string** READONLY

The Active Directory domain name for the submit user if Authentication Services is configured and the client is able to determine the domain name. Otherwise this variable is set to an empty string.

Related Topics

samaccount

env

Description

Type **list** READONLY

`env` contains the list of environment variables configured in the environment where the submit user submitted the request.

false

Description

Type **integer** READONLY

`false` contains the constant value 0.

Related Topics

true

gid

Description

Type **integer** READONLY

`gid` contains the Group ID of the submitting user's primary group on the client host.

Related Topics

uid

group

group

Description

Type **string** READONLY

`group` contains the name of user's primary group.

Related Topics

groups

groups

Description

Type **string** READONLY

`groups` contains the list all groups in which the user is a member.

Related Topics

group

host

Description

Type **string** READONLY

`host` identifies the host name where the user has requested to run the command.

hour

Description

Type **integer** READONLY

`hour` contains the hour the request was submitted (0 – 23).

Related Topics

dayname

minute

day

month

year

time

date

masterhost

Description

Type **string** READONLY

`masterhost` contains the host name of the host running `pmmasterd`.

masterversion

Description

Type **string** READONLY

`masterversion` contains the description of Safeguard policy server host.

minute

Description

Type **integer** READONLY

`minute` contains the minute the request was submitted (0-59).

Related Topics

dayname

hour

day

month

year

time

date

month

Description

Type **integer** READONLY

`month` contains the month number the request was submitted (0-11).

Related Topics

dayname

minute

hour

day

year

time

date

nice

Description

Type **integer** READONLY

`nice` contains the value of the submit user session's `nice` value, that controls the execution priority. For more information, see the `nice` man pages.

nodename

Description

Type **string** READONLY

`nodename` contains the host name of the client host.

Related Topics

submithost

optarg

Description

Type **string** READONLY

`optarg` contains the parameter for the last argument or, if the option takes no argument, an empty string . Use with `getopt` functions.

opterr

Description

Type **boolean** READONLY

`opterr` determines whether to show errors from `getopt` functions.

optind

Description

Type **integer** READONLY

`optind` contains the current argument list index. Use with `getopt` functions.

optopt

Description

Type **string** READONLY

`optopt` contains the letter of the last option that had an issue. Use with `getopt` functions.

optreset

Description

Type **boolean** READONLY

When set to True, `optreset` restarts the `getopt` functions from the beginning. The next time a user calls a `getopt` function, `optind` will be set to 1.

optstrictparameters

Description

Type **boolean** READONLY

The `getopt_long()` function provides specific argument parameters. Arguments with optional parameters are accepted only when entered in the format `--argument=parameter`. For `getopt_long()` to recognize non-compliant forms, such as `--argument parameter`, set `optstrictparameters` to False.

pid

Description

Type **integer** READONLY

`pid` contains the process ID number of the `pmmasterd` process.

pmclient_type

Description

Type **integer** READONLY

The client type (pmrun or sudo) of the Safeguard request.

Related Topics

pmclient_type_pmrun

pmclient_type_sudo

pmclient_type_pmrun

Description

Type **integer** READONLY

Read-only constant for sudo type clients. You can compare `pmclient_type_pmrun` to `pmclient_type` to determine if the request was sent from a Privilege Manager for Unix client including the `pmrun` command.

Related Topics

pmclient_type

pmclient_type_sudo

pmclient_type_sudo

Description

Type **integer** READONLY

Read-only constant for sudo type clients. You can compare `pmclient_type_sudo` to `pmclient_type` to determine if the request was sent from a Sudo Plugin client.

Related Topics

pmclient_type

pmclient_type_pmrun

pmversion

Description

Type **string** READONLY

`pmversion` contains the Safeguard version and build number.

ptyflags

Description

Type **string** READONLY

`ptyflags` contains a `bitmask` indicating the `ptyflags` set from the submit user's environment. If set, the following bits indicate:

```
Bit 0: stdin is open
Bit 1: stdout is open
Bit 2: stderr is open
Bit 3: command was run in pipe mode
Bit 4: stdin is from a socket
Bit 5: command to be run using nohup
```

requestlocal

Description

Type **integer** READONLY

Indicates if the request is local. `requestlocal` is always set to `true` for sudo commands.

requestuser

Description

Type **string** READONLY

`requestuser` is initialized to the selected user name if you select the `sudo -u` option. It is a request to set the `runuser` for the session to the selected user name. The administrator can decide whether to honor the request in the policy file. By default, this variable is set to the value of the user variable.

rlimit_as

Description

Type **string** READ ONLY

The `rlimit_as` variable controls the maximum memory that is available to a process.

rlimit_core

Description

Type **string** READ ONLY

The `rlimit_core` variable controls the maximum size of a core file.

rlimit_cpu

Description

Type **string** READ ONLY

The `rlimit_cpu` variable controls the maximum size CPU time of a process.

rlimit_data

Description

Type **string** READ ONLY

The `rlimit_data` variable controls the maximum size of data segment of a process.

rlimit_fsize

Description

Type **string** READ ONLY

The `rlimit_fsize` variable controls the maximum size of a file.

rlimit_locks

Description

Type **string** READ ONLY

The `rlimit_locks` variable control the maximum number of file locks for a process.

rlimit_memlock

Description

Type **string** READ ONLY

The `rlimit_memlock` variable controls the maximum number of bytes of virtual memory that can be locked.

rlimit_nofile

Description

Type **string** READ ONLY

The `rlimit_nofile` variable controls the maximum number of files a user may have open at a given time.

rlimit_nproc

Description

Type **string** READ ONLY

The `rlimit_nproc` variable controls the maximum number of processes a user may run at a given time.

rlimit_rss

Description

Type **string** READ ONLY

The `rlimit_rss` variable controls the maximum size of the resident set (number of virtual pages resident at a given time) of a process.

rlimit_stack

Description

Type **string** READ ONLY

The `rlimit_stack` variable controls the maximum size of the process stack.

samaccount

Description

Type **string** READONLY

The user's sAMAccountName for the submit user if Authentication Services is configured and the client is able to determine the sAMAccountName. Otherwise this variable is set to an empty string.

Related Topics

domainname

selinux

Description

Type **boolean** READONLY

`selinux` detects whether the client running pmrun or sudo is within an SELinux environment.

If SELinux is enabled on the client or policy host machine, it is True. If disabled, it is False.

status

Description

Type **integer** READONLY

`status` contains the exit status of the most recent command run by the system function.

submithost

Description

Type **string** READONLY

`submithost` contains the name of the host where the request was submitted.

Related Topics

host

submithostip

Description

Type **string** READONLY

`submithostip` contains the IP address of the host where a request was submitted.

Related Topics

submithost

thishost

Description

Type **string** READONLY

The value of the `thishost` setting in the `pm.settings` file on the client. If you do not specify the `thishost` setting or if the client cannot resolve `thishost` to an IP address configured on the client, the variable remains undefined.

Related Topics

host

submithost

time

Description

Type **string** READONLY

`time` contains the time the request was submitted in the form HH:MM:SS.

Related Topics

dayname

minute

hour

day

month

year

date

true

Description

Type **integer** READONLY

`true` is a read-only constant with a value of `1`.

Related Topics

false

ttyname

Description

Type **string** READONLY

`ttyname` contains the name of the TTY device from which the user submitted a request.

tzname

Description

Type **string** READONLY

Description

The time zone variable, `tzname`, contains the name of the time zone on the server at the time the event was read from the event log by `pmlog`. The time zone may be overridden using the `TZ` environment variable when running `pmlog`.

Note that `tzname` is accessible from `pmlog` but not in the policy script evaluation.

Example

```
# pmlog -p `sprintf("%s %s %s, %s, %s", date, time, tzname, event,
uniqueid)’
2013-03-14 10:51:59 MDT, Accept, 0b1c7ff3447ac074b4795be2dcd59f6429c8624b
2013-03-14 10:51:59 MDT, Accept, a6cfad1ba6eb64bf9a17d5295b2bb29daa7fbb33
2013-03-14 10:51:59 MDT, Accept, fa742929679bc6c88eadd25ff85d75361f1d28b2
2013-03-14 10:51:59 MDT, Accept, 97ffdb433819c5feab6ec26b528f60dfb18c3d34
2013-03-15 07:02:47 MDT, Accept, d84ac9052265912eb13d32f80584d1ae097e4ce5
2013-03-19 09:41:59 MDT, Accept, b228110f32525c2092d2a46d0327e55f2dfc1d39
```

The actual values may vary by platform. In this sample output, the value of `tzname` is "MDT".

The following example shows the use of the `TZ` variable acting on the output:

```
TZ=Europe/Paris pmlog -p `sprintf( "%s %s %s, %s", date, time, tzname,
event )’
2013-03-14 17:51:59 CET, Accept, 0b1c7ff3447ac074b4795be2dcd59f6429c8624b
2013-03-14 17:51:59 CET, Accept, a6cfad1ba6eb64bf9a17d5295b2bb29daa7fbb33
2013-03-14 17:51:59 CET, Accept, fa742929679bc6c88eadd25ff85d75361f1d28b2
2013-03-14 17:51:59 CET, Accept, 97ffdb433819c5feab6ec26b528f60dfb18c3d34
2013-03-15 14:02:47 CET, Accept, d84ac9052265912eb13d32f80584d1ae097e4ce5
2013-03-19 16:41:59 CET, Accept, b228110f32525c2092d2a46d0327e55f2dfc1d39
```

Related Topics

date

time

uid

Description

Type **integer** READONLY

`uid` contains the user ID of the submitting user on the sudo host.

Related Topics

gid

group

umask

Description

Type **integer** READONLY

`umask` contains the value of the submit user's `umask` value. See the `umask` man page for details.

unameclient

Description

Type **list** READONLY

`unameclient` contains the system `uname` information from the client host. This information corresponds to the list returned by `uname`. For example:

- operating system name
- nodename
- operating system release level
- operating system version
- machine hardware name

uniqueid

Description

Type **string** READONLY

`uniqueid` is a 12-character string identifying a session. This is guaranteed to be unique on one policy server machine.

user

Description

Type **string** READONLY

`user` containts the submit user's login name.

year

Description

Type **integer** READONLY

`year` contains the year in which the request was submitted in the format YY.

Related Topics

dayname

minute

hour

day

month

date

time

Global output variables

The following predefined global variables are initialized from the submit user's environment.They can be affected by the policy file.

Table 11: Global output variables

Variable	Data Type	Description
disable_exec	integer	Specifies whether to prevent the `runcommand` process from executing new processes.
eventlog	string	Pathname of the audit log.
iolog	string	Pathname of the keystroke log.
logstderr	integer	Specifies whether to keystroke log `stderr` messages.
logstdin	integer	Specifies whether to keystroke log `stdin` messages.
logstdout	integer	Specifies whether to keystroke log `stdout` messages.
runargv	list	List of arguments for the request.
runchroot	string	Requests the command to run with a specified `root` directory.
runcksum	string	Identifies a checksum to use to verify against the `runcommand`.
runclienthost	string	A modifiable copy of the `clienhost` input variable.
runcommand	string	Full pathname of the request.
runconfirmuser	string	Specifies whether the agent should request the `runuser` to authenticate before executing the `runcommand`.
runcwd	string	Working directory to set for the request.
runenablerlimits	boolean	Lets you use `runrlimit` variables on the run host.
runenv	list	List of environment variables to set for the request.
rungroup	string	Primary group to set for the request.
rungroups	list	List of secondary groups to set for the request.
runhost	string	Host on which to run the request.
runnice	integer	Nice value to apply for the request.
runpaths	list	A list of permitted paths for commands.
runptyflags	string	`Pty` flags to apply for the request.
runrlimit_as	string	Controls the maximum memory that is available to a process.
runrlimit_core	string	Controls the maximum size of a core file.
runrlimit_cpu	string	Controls the maximum size CPU time of a process.

Variable	Data Type	Description
runrlimit_data	string	Controls the maximum size of data segment of a process.
runrlimit_fsize	string	Controls the maximum size of a file.
runrlimit_locks	string	Control the maximum number of file locks for a process.
runrlimit_memlock	string	Controls the maximum number of bytes of virtual memory that can be locked.
runrlimit_nofile	string	Controls the maximum number of files a user may have open at a given time.
runrlimit_nproc	string	Controls the maximum number of processes a user may run at a given time.
runrlimit_rss	string	Controls the maximum size of the resident set (number of virtual pages resident at a given time) of a process.
runrlimit_stack	string	Controls the maximum size of the process stack.
runtimeout	integer	Specifies the number of seconds of idle time before ending the session.
runumask	integer	`Umask` value to apply for the request.
runuser	string	User to run the request.
runutmpuser	string	`Utmp` user to use when logging to `utmp`.
subprocuser	string	User name to run subprocesses of the policy server master daemon.
tmplogdir	string	Directory used for temporary storage of I/O log files if a remote log host is specified in `iologhost`.

disable_exec

Description

Type **integer** READ/WRITE

Use `disable_exec` to prevent the `runcommand` process from executing new UNIX processes. For example, you can prevent a `vi` session from executing shell commands. This variable is only supported if the underlying operating system supports the `noexec` feature; that is, Linux, Solaris, HP-UX, and AIX. If set to `true(1)`, Safeguard sets the `LD_PRELOAD` environment variable, which causes the `runcommand` to be loaded with a Safeguard library that overrides the system `exec` functions, and thus prevents the `runcommand` from using `exec` to create a new process.

eventlog

Description

Type **string** READ/WRITE

`eventlog` contains the full pathname of the file in which audit events are logged. The default pathname is `/var/opt/quest/qpm4u/pmevents.db`.

Related Topics

event

Event logging

iolog

Description

Type **string** READ/WRITE

`iolog` is the full path name of the keystroke log file in which input, output, and error output is logged.

logstderr

Description

Type **integer** READ/WRITE

Set `logstderr` to `true` to enable keystroke logging of `stderr` output produced during the session. The default value is `true`.

Related Topics

logstdin

logstdout

logstdin

Description

Type **integer** READ/WRITE

Set `logstdin` to `true` to enable keystroke logging of `stdin` input produced during the session. The default value is `true`.

Related Topics

logstderr

logstdout

logstdout

Description

Type **integer** READ/WRITE

Set `logstdout` to `true` to enable keystroke logging of `stdout` output produced during the session. The default value is `true`.

Related Topics

logstderr

logstdin

runrlimit_as

Description

Type **string** WRITABLE

`runrlimit_as` is a modifiable copy of the `rlimit_as` input variable. It controls the maximum memory that is available to a process.

Related Topics

rlimit_as

runrlimit_core

Description

Type **string** WRITABLE

`runrlimit_core` is a modifiable copy of the `rlimit_core` input variable. It controls the maximum size of a core file.

Related Topics

rlimit_core

runrlimit_cpu

Description

Type **string** WRITABLE

`runrlimit_cpu` is a modifiable copy of the `rlimit_cpu` input variable. It controls the maximum size CPU time of a process.

Related Topics

rlimit_cpu

runrlimit_data

Description

Type **string** WRITABLE

`runrlimit_data` is a modifiable copy of the `rlimit_data` input variable. It controls the maximum size of the data segment of a process.

Related Topics

rlimit_data

runrlimit_fsize

Description

Type **string** WRITABLE

`runrlimit_fsize` is a modifiable copy of the `rlimit_fsize` input variable. It controls the maximum size of the data segment of a file.

Related Topics

rlimit_fsize

runrlimit_locks

Description

Type **string** WRITABLE

`runrlimit_locks` is a modifiable copy of the `rlimit_locks` input variable. It controls the maximum number of file locks for a process.

Related Topics

rlimit_locks

runrlimit_memlock

Description

Type **string** WRITABLE

`runrlimit_memlock` is a modifiable copy of the `rlimit_memlock` input variable. It controls the maximum number of bytes of virtual memory that can be locked.

Related Topics

rlimit_memlock

runrlimit_nofile

Description

Type **string** WRITABLE

`runrlimit_nofile` is a modifiable copy of the `rlimit_nofile` input variable. It controls the maximum number of files a user may have open at a given time.

Related Topics

rlimit_nofile

runrlimit_nproc

Description

Type **string** WRITABLE

`runrlimit_nproc` is a modifiable copy of the `rlimit_nproc` input variable. It controls the maximum number of processes a user may run at a given time.

Related Topics

rlimit_nproc

runrlimit_rss

Description

Type **string** WRITABLE

`runrlimit_rss` is a modifiable copy of the `rlimit_rss` input variable. It controls the maximum size of the resident set (number of virtual pages resident as a given time) of a process.

Related Topics

rlimit_rss

runrlimit_stack

Description

Type **string** WRITABLE

`runrlimit_stack` is a modifiable copy of the `rlimit_stack` input variable. It controls the maximum size of the process stack.

Related Topics

rlimit_stack

runtimeout

Description

Type **string** READ/WRITE

`runtimeout` specifies the number of seconds of idle time allowed before the session is closed.

Example

```
# close the session if the user is idle for 5 minutes
runtimeout=300;
```

runumask

Description

Type **integer** READ/WRITE

`runumask` is a modifiable copy of the `umask` input variable. Specifies the `umask` filter which determines file permissions for files created during execution of the `runcommand`.

Example

```
trustedusers = {"jamie", "cory", "robyn"};
 if (user in trustedusers )
 {
      runumask=066;
 }
```

Related Topics

umask

runuser

Description

Type **string** READ/WRITE

`runuser` is a modifiable copy of the `user` input variable. Specifies the user name that `pmlocald` uses when initializing the runtime environment for the `runcommand`.

Example

```
if ( (user == "apache") && (command == "admin.cgi") )
{
     runuser="root";
}
```

Related Topics

user

runutmpuser

Description

Type **string** READ/WRITE

`runutmpuser` specifies the login name of the user that will be used when updating the UNIX `utmp` and `wtmp` files when the request runs.

Example

```
if ( user == "djv" )
{
     runutmpuser="dave";
}
```

subprocuser

Description

Type **string** READ/WRITE

`subprocuser` is the user name used to run any subprocesses of `pmmasterd` such as, when running the system function. The default value is "root".

Example

```
subprocuser="appl_user";
 cfile=system("find  /home/applhome  -name  customprofile.txt");
 if  (status  ==  0)
 {
      print(readfile(cfile));
 }
```

Global event log variables

The following predefined global variables appear only in the audit (event) log. They are not available for use in the policy file, as they are set by `pmlocald` during the `runcommand` session.They are set by the Sudo Plugin during the `runcommand` session. Use `pmlog` to view them.

Table 12: Global event log variables

Variable	Data Type	Description
alertdate	string	Date on which the alert was raised.
alerttime	string	Time at which the alert was raised.
event	string	Type of event.
exitdate	string	Date on which the finish event was logged.
exitstatus	string	Exit status of the request
exittime	string	Exit time of the request.

event

Description

Type **string** READONLY

`event` identifies the type of event logged by the policy server process. An event is logged when the policy server accepts or rejects a command.

Possible values are:

- Accept
- Reject
- Finish

This value is saved in the event log and can be viewed using `pmlog`.

Example

```
#Display all accepted events from the audit log
pmlog -c 'event == "Accept"'
```

Related Topics

eventlog

exitdate

Description

Type **string** READONLY

`exitdate` is the date the requested command finished running. This is saved in the event log when the session exits, and can be viewed using `pmlog`.

Example

```
#Display all events that finished on 15 january 2009
pmlog -c 'exitdate == "2009/01/15"'
```

Related Topics

exitstatus

exittime

exitstatus

Description

Type **string** READONLY

`exitstatus` contains the exit status of the `runcommand`. This variable is not available for use in the policy file. It is logged in the "Finish" event by the Sudo Plugin when the session ends.

Example

```
#Display all sh commands that failed to complete successfully
pmlog -c 'runcommand == "sh" && exitstatus != "Command finished with
exit status 0"'
```

Related Topics

exitdate

exittime

exittime

Description

```
Type string READONLY
```

`exittime` is the time the requested command finished running (HH:MM:SS)

Example

```
#display all commands that finished after 6pm
pmlog -c 'exittime > "18:00:00"'
```

Related Topics

exitstatus

exitdate

PM settings variables

This section describes the settings and parameters used by Safeguard. These settings are stored on each host in the `/etc/opt/quest/qpm4u/pm.settings` file which contains a list of settings, one per line, in the form: settingName value1 [value2 [... value*n*]].

You can modify these policy server configuration settings using the configuration script initialized by either the `pmsrvconfig` or `pmjoin_plugin` commands; or you can modify the `pm.settings` file manually. See Configuring the Safeguard for Sudo Primary Policy Server on page 19 for details about running the configuration script.

If you manually change the `pm.settings` file, restart the `pmserviced` and/or `pmloadcheck` daemons in order for the changes to take effect.

The following table describes each of the `pm.settings` variables:

Defaults may differ depending on the platform you are configuring and whether you are configuring a policy server or Sudo Plugin. Many of these settings will not have a default value.

The variables are not case sensitive.

Table 13: Variables: pm.settings

Variable	Data type	Description
auditsrvCAbundle	string	The path to a certificate authority bundle file, in PEM format, to use instead of the system's default certificate authority database when doing TLS authentication. Example: `/etc/ssl/sudo/ca.bundle.pem`
auditsrvCert	string	The path to the policy server's certificate file, in PEM format. Used for TLS only.

Variable	Data type	Description
		Example: `/etc/ssl/sudo/qpm_qpmdevel1.cert.pem`
auditsrvEnabled	boolean (YES/NO)	Specifies if audit server logging is on or off. The default is NO. For more information, see Audit server logging on
auditsrvEnforced	boolean (YES/NO)	If YES, the audit server connection failures will be fatal. If NO, the audit log will be collected encrypted on the file system.and sent again to the server if it comes back.
auditsrvHosts	list of host ports of the audit sever	The host:port of the audit server. Host can be an `ipv4/ipv6/hostname`. Multiple hosts need to be separated by comma. Example: `qpmdevel1.qpmdomain:30344,127.0.0.1:30344`
auditsrvKeepalive	boolean (YES/NO)	Select whether keepalive is enabled on the connection.
auditsrvLocaliologs	boolean (YES/NO)	If YES, old `pmlog` io logs are also written. if NO, io logs are only stored by the sudo log server.
auditsrvLogdir	string	If `auditsrvEnforced` is NO, this is the log directory where audit logs get saved temporarily until they can be sent successfully. Example: `/var/opt/quest/qpm4u/auditserver`
auditsrvPkey	string	The path to the private key of the policy server, in PEM format. Used for TLS only. Example: `/etc/ssl/sudo/qpm_qpmdevel1.key.pem`
auditsrvPSpaceMB	boolean (YES/NO)	The minimum amount of disk space needed before starting to write an audit trail to the temporary storage. This is to prevent disk space filled up. If the disk space is lower, the policy server will reject the connections, just like if it was in "enforced" mode.

Variable	Data type	Description
auditsrvTimeout	integer	The connection timeout in seconds. 3 seconds is recommended.
auditsrvTLS	boolean (YES/NO)	If YES, the communication with all the servers will use TLS. Specifying a key is required in this case.
auditsrvTLSCheckpeer	boolean (YES/NO)	If YES, client certificates will be validated by the server; clients without a valid certificate will be unable to connect. If NO, no validation of client certificates will be performed. If true and client certificates are created using a private certificate authority, the `tls_cacert` setting must be set to a CA bundle that contains the CA certificate used to generate the client certificate. The default value is NO.
auditsrvTLSVerify	boolean (YES/NO);	If YES, the server certificate will be verified at startup and clients will authenticate the server by verifying its certificate and identity. If NO, no verification is performed of the server certificate by the server or the client. When using self-signed certificates without a certificate uthority, this setting should be set to NO. The default value is YES.
certificates	boolean (YES/NO)	Specifies whether certificates are enabled. To enable configurable certification, add the following statement to the `/etc/opt/quest/qpm4u/pm.settings` file on each host: `certificates yes`.
checksumtype	string	Specifies standard or MD5 checksum types for use with `pmsum` program.
clients	list of hostnames	Identifies hosts for which remote access functions are allowed. Only required if one policy server needs to retrieve remote information from another policy server that does not normally accept requests from it. For more information, see Central logging with Privilege Manager for Unix.

Variable	Data type	Description
clientverify	string	Identifies the level of host name verification applied by the policy server host to the submit host name. The verification ensures that the incoming IP address resolves (on the primary policy server) to the same host name as presented by the submit host. Valid values are: • **none**: No verification performed. • **yes**: If a host name is presented for verification by the runclient it will be verified. • **All**: The policy server will only accept a request from a client if the host name is verified. Default: NONE
encryption	string	Identifies the encryption type. You must use the same encryption setting on all hosts in your system. Valid values are: • AES • DES • TripleDES Default: AES
eventlogqueue	string	Directory used by `pmmasterd` and `pmlogsrvd` where event data is temporarily queued prior to being written to the event log database. Default: `/var/opt/quest/qpm4u/evcache`
EventQueueFlush	integer	Tells pmlogsrvd how often to reopen the db (in minutes) flushing the data. Default: 0, in which case pmlogsrvd will keep the db open while the service is running.
EventQueueProcessLimit	integer	Specifies the number of cached events that will be processed at a time; this limits the memory use in pmlogsrvd. Default: 0, in which case pmlogsrvd will not apply a limit.
facility	string	Sets the SYSLOG facility name to use when

Variable	Data type	Description
		logging a message to the `syslog` file. Valid values are: • LOG_AUTH • LOG_CRON • LOG_DAEMON • LOG_KERN • LOG_LOCAL0 through LOG_LOCAL7 • LOG_LPR • LOG_MAIL • LOG_NEWS • LOG_USER • LOG_UUCP Default: LOG_AUTH, if the platform defines LOG_AUTH; otherwise the default is 0 (zero).
failovertimeout	integer	Sets the timeout in seconds before a connection attempt to a policy server is abandoned and the client fails over to the next policy server in the list. This setting also affects the timeout for the client and agent. Default: 10 seconds. If omitted from `pm.settings`, default is 180 seconds.
fwexternalhosts	list	Identifies a list of hosts to use a different range of source ports, identified by the `openreservedport` and `opennonreserved` port settings.
getpasswordfromrun	boolean (YES/NO)	Determines whether authentication is performed on the policy server or the client when a `getuserpasswd()` or `getgrouppasswd()` function is called from the policy file. If set to `yes`, the authentication is performed on the client. This variable also affects the user information functions: `getfullname()`, `getgroup()`, `getgroups()`, `gethome()`, and `getshell()`. If set to **yes** in the policy server's `pm.settings` file, these functions retrieve user information

Variable	Data type	Description
		from the client host. Default: NO
handshake	boolean (YES/NO)	Enables the encryption negotiation handshake. This allows a policy server to support clients running different levels of encryption. Default: NO
kerberos	boolean (YES/NO)	Enables or disables Kerberos. Default: NO
keytab	string	Sets the path to the Kerberos keytab file. Default: `/etc/opt/quest/vas/host.keytab`
krb5rcache	string	Sets the path to the Kerberos cache. Default: `/var/tmp`
krbconf	string	Sets the path to the Kerberos configuration file. Default: `/etc/opt/quest/vas/vas.conf`
libldap	string	Specifies the pathname to use for the LDAP library. No default value.
localport	integer	Sets the TCP/IP port to use for `pmlocald`. Default: 12346
lprincipal	string	Sets the service principal name to use for the agent. Default: pmlocald
masterport	integer	Specifies the TCP/IP port to use for `pmmasterd`. Default: 12345
masters	list	Identifies a list of policy server hosts to which a client can submit requests for authorization, and from which an agent can accept authorized requests. This can contain host names or netgroups. No default value.
maxofflinelogs	integer	Sets the maximum number of offline keystroke or event logs that can be

Variable	Data type	Description
		transferred to a policy server in a single transaction. If defined on the policy server, `pmmasterd` on the server only accepts that number of offline logs from a client in a single request. If configured on a plugin, the plugin only tries to send that number of logs at a time. No default value.
mprincipal	string	Sets the Kerberos service principal name to use for the policy server. Default: host
nicevalue	integer	Sets the execution priority level for Safeguard processes. Default: 0
offlinetimeout	integer	Sets the timeout in milliseconds before an off-line policy evaluation occurs on a Sudo Plugin host. Default: 1500 (1.5 seconds) Setting `offlineTimeout` to 0 in the `pm.settings` file, forces the cache service to always perform offline (local-only) policy evaluation for sudo requests.
opennonreserveportrange	integer integer	Specifies a range of non-reserved ports to use as source ports when connecting to a host in the `fwexternalhosts` list. No default value.
openreserveportrange	integer integer	Specifies a range of reserved ports to use as source ports when connecting to a host in the `fwexternalhosts` list. No default value.
pmclientdenabled	boolean (YES/NO)	Flag that enables the `pmclientd` daemon.
pmclientdopts	string	Sets the options for the `pmclientd` daemon.
pmlocaldenabled	boolean (YES/NO)	Flag that enables the `pmlocald` daemon.
pmlocaldlog	string	Sets the path for the agent error log. Default: `/var/adm/pmlocald.log` or

Variable	Data type	Description
		`/var/log/pmlocald.log` depending on the platform. For more information, see Local logging on page 53..
pmlocaldopts	string	Sets the options for the `pmlocald` daemon.
pmloggroup	string	Specifies the group ownership for `iolog` and `eventlogs`. Default: pmlog
pmlogsrvlog	string	Identifies the log used by the `pmlogsrvd` daemon.
pmmasterdenabled	boolean (YES/NO)	Flag that enables the `pmmasterd` daemon. Default: YES
pmmasterdlog	string	Sets the path for the master error log. Default: `/var/adm/pmmasterd.log` or `/var/log/pmmasterd.log` depending on the platform. For more information, see Local logging on page 53..
pmmasterdopts	string	Sets the options for the `pmmasterd` daemon. Default: -ar
pmrunlog	string	Sets the path for the client error log. Default: `/var/adm/pmrun.log` or `/var/log/pmrun.log` depending on platform. For more information, see Local logging on page 53..
pmservicedlog	string	Identifies the log used by the `pmserviced` daemon. Default: `/var/log/pmserviced.log`
pmtunneldenabled	boolean (YES/NO)	Flag that enables the `pmtunneld` daemon.
pmtunneldopts	string	Sets the options for the `pmtunneld` daemon.
policydir	string	Sets the directory in which to search for policy files Default: `/etc/opt/quest/qpm4u/policy`
policyfile	string	Sets the main policy filename.

Variable	Data type	Description
		Default: `pm.conf`
policymode	string	Specifies the type of security policy to use, pmpolicy or Sudo. Default: sudo
reconnectagent	boolean (YES/NO)	Allows backwards compatibility with older agents on a policy server. Settings on policy server and agents must match. Default: NO
reconnectclient	boolean (YES/NO)	Allows backwards compatibility with older clients on a policy server. Settings on policy server and client must match. Default: NO
selecthostrandom	boolean (YES/NO)	Set to `yes` to attempt connections to the list of policy servers in random order. Set to `no` to attempt connections to the list of policy servers in the order listed in `pm.settings`. Default: YES
setnonreserveportrange	integer integer	Specifies a range of non-reserved ports to use as source ports by the client and agent. • Minimum non-reserved port is 1024. • Maximum non-reserved port is 31024. The full range for non-reserved ports is 1024 to 65535.
setreserveportrange	integer integer	Specifies a range of reserved ports to use as source ports by the client when making a connection to the policy server. • Minimum reserved port is 600. • Maximum reserved port is 1023. The full range for reserved ports is 600 to 1023.
setutmp	boolean (YES/NO)	Specifies whether `pmpluginadds` a `utmp` entry for the request. Default: YES
shortnames	boolean	Enables or disables short names usage.

Variable	Data type	Description
	(YES/NO)	Setting `shortnames` to `yes` allows the use of short (non-fully qualified) host names. If set to `no`, then the Safeguard components will attempt to resolve all host names to a fully qualified host name. Default: YES
sudoersfile	string	Sets the path to the sudoers policy file, if using the Sudo policy type. Default: /etc/opt/quest/qpm4u/policy/sudoers
sudoersgid	integer	Sets the group ownership of the Sudoers policy, if using the Sudo policy type. Default: 0
sudoersmode	integer	Sets the UNIX file permissions of the Sudoers policy, if using the sudo policy type. Specify it as a four-digit octal number (containing only digits 0-7) to determine the user's file access rights (read, write, execute). Default: 0400
sudoersuid	integer	Sets the user ownership of the Sudoers policy. Default: 0
syslog	boolean (YES/NO)	Set to `yes` to send error messages to the syslog file as well as to the Safeguard error log. Default: YES For more information, see Local logging on page 53..
thishost	string	Sets the client's host name to use for verification. Specifying a `thishost` setting causes the Privilege Manager components to bind network requests to the specified host name or IP address. If you set `thishost` to the underscore character (_), requests bind to the host's primary host name. No default value.
tunnelport	integer	Sets the TCP/IP port to use for the `pmtunneld` daemon.

Variable	Data type	Description
		Default: 12347 For more information, see Configuring pmtunneld.
tunnelrunhosts	list	Identifies the hosts on the other side of a firewall. No default value. For full details of how to configure your system across a firewall, see Configuring Firewalls.
utmpuser	string	Specifies which user name `pmplugin` logs to the `utmp` entry. Valid values are: • `submituser` • `runuser` To log an entry to `utmp`, specify "setutmp yes". These settings only take effect if the sudoers policy allocates a pty. A `pseudo-tty` is allocated by sudo when the `log_input`, `log_output` or `use_pty` flags are enabled in sudoers policy. Default: `submituser`
validmasters	list	Identifies a list of policy servers that can be identified using the `pmrun -m <master>` option, but that will not be used when you run a normal `pmrun` command. This is useful for testing connections to a policy server before bringing it on line. No default value.

Appendix B

Safeguard programs

This section describes each of the Safeguard programs and their options. The following table indicates which Safeguard component installs each program.

Table 14: Privilege Manager programs

Name	Description	Server	Agent	Sudo
pmauditsrv	Verifies that the configured audit servers are accessible and configured properly and exchanges a "hello" message with the server. If the audit server is not accessible, stores the events and keystroke (IO) logs temporarily offline and sent to the audit server when it is available.	X	N/A	N/A
pmcheck	Verifies the syntax of a policy file.	X	N/A	X
pmgit	The `pmgit` utility is used to configure Git policy management for Privilege Manager for Unix.	X	X	N/A
pmjoin_plugin	Joins a Sudo Plugin to the specified policy server. Joining configures the remote host to communicate with the servers in the group.	X	N/A	X
pmkey	Generates and installs configurable certificates.	X	X	X
pmlicense	Displays current license information and allows you to update a license (an expired one or a temporary one before it expires) or create a new one.	X	N/A	N/A

Name	Description	Server	Agent	Sudo
pmloadcheck	Controls load balancing and failover for connections made from the host to the configured policy servers.	X	X	N/A
pmlog	Displays entries in a Privilege Manager for Unix event log.	X	N/A	N/A
pmlogadm	Manages encryption options on the event log.	X	N/A	N/A
pmlogsearch	Searches all logs in a policy group based on specified criteria.	X	N/A	N/A
pmlogsrvd	The Privilege Manager for Unix log access daemon, the service responsible for committing events to the Privilege Manager for Unix event log and managing the database storage used by the event log.	X		
pmlogxfer	Transfers event logs and I/O logs after an off-line policy evaluation has occurred. `pmlogxfer` is initiated by `pmloadcheck` when there are log files queued for transfer from a Sudo Plugin host to the server.	N/A	N/A	X
pmmasterd	The Privilege Manager for Unix Master daemon which examines each user request and either accepts or rejects it based upon information in the Privilege Manager configuration file. You can have multiple `pmmasterd` daemons on the network to avoid having a single point of failure.	X	N/A	X
pmplugininfo	Displays information about the policy server group that the Sudo Plugin host has joined.	X	N/A	X
pmpluginloadcheck	A daemon that runs on each Sudo Plugin host and controls load balancing and failover for connections made from the host to the configured policy servers.	X	N/A	X

Name	Description	Server	Agent	Sudo
pmpolicy	A command-line utility for managing the Privilege Manager for Unix security policy. This utility checks out the current version, checks in an updated version, and reports on the repository.	X	N/A	N/A
pmpolicyplugin	Displays the revision status of the cached security policy on a Sudo Plugin host; allows you to request an update from the central repository.	N/A	N/A	X
pmpoljoin_plugin	Adjunct program to the `pmjoin_plugin` script. `pmpoljoin_plugin` is called by the `pmjoin_plugin` script when configuring a Sudo Plugin host to setup up the required read-only access to the policy repository, so that the client can operate in off-line mode.	N/A	N/A	X
pmpolsrvconfig	Configures (or unconfigures) a primary or secondary policy server. Allows you to grant a user access to a repository.	X	N/A	N/A
pmremlog	Provides a wrapper for the `pmlog` and `pmreplay` utilities to access the event (audit) and keystroke (I/O) logs on any server in the policy group.	X	N/A	N/A
pmreplay	Replays an I/O log file allowing you to review what happened during a previous privileged session.	X	N/A	N/A
pmresolvehost	Verifies the host name or IP resolution for the local host or a selected host.	X	X	X
pmserviced	The Privilege Manager for Unix Service daemon listens on the configured ports for incoming connections for the Privilege Manager for Unix daemons. `pmserviced` uses options in `pm.settings` to determine the	X	X	X

Name	Description	Server	Agent	Sudo
	daemons to run, the ports to use, and the command line options to use for each daemon.			
pmsrvcheck	Checks the Privilege Manager for Unix policy server configuration to ensure it is setup properly.	X	N/A	N/A
pmsrvconfig	Configures a primary or secondary policy server.	X	N/A	N/A
pmsrvinfo	Verifies the policy server configuration.	X	N/A	N/A
pmsum	Generates a simple checksum of a binary.	X	N/A	N/A
pmsysid	Displays the Privilege Manager for Unix system ID.	X	X	X

pmauditsrv

Syntax

```
$ pmauditsrv -h
Usage: pmauditsrv [-h] [-v] [-z on|off]
Usage: pmauditsrv check|send [ -o <serverlist> ] [ -b <ca_bundle_file> ] [ -k
<privatekey_file> ] [ -c <certificate_file> ] [ -s ]
```

Description

Use `pmauditsrv` for the following:

- `pmauditsrv` verifies that the configured audit servers are accessible and configured properly. This includes verifying that certificates and keys are configured properly for TLS communication, if enabled. `pmauditsrv` exchanges a "hello" message with the server.
- When the policy server is configured for "not enforced mode" and the audit server is not accessible, `pmauditsrv` can be used to store the event logs and keystroke (IO) logs temporarily offline. `pmauditsrv` sends the logs to the audit server once it is available. If the connection to the audit server was broken in the middle of the command run and the log is a partial log, the log will be sent to the same server that received the first part of the message. Logs which are not partial logs are sent to the audit servers according to the actual configuration. Changing the auditserver configurations can solve transferring full but not partial logs.

By default, the `pmloadcheck` program executes `pmauditsrv` in every 30 minutes to transfer any audit trail files found in the configured cache directory to the audit server. If the file can not be processed (for example, the file is corrupt), `pmauditsrv` moves the file into a subdirectory (quarantine).

`pmauditsrv` can be called manually for troubleshooting an issue.

With command 'check' .B, `pmauditsrv` can be also used to check connection to the configured audit servers or the server specified with command line arguments.

Errors logs are stored in `/var/log/pmmasterd.log`.

Options

`pmauditsrv` has the following options.

Table 15: Options: `pmauditsrv`

Option	Description
-h	Display a help usage information and exit.
-v	Display the version number of the `pmauditsrv` program and exit.
-z on | off	Turn debug tracing on or off, then exit.
-o <serverlist>	Specify audit servers. Format: `addr1:port1..addrn:portn` where `addr` is either IP or hostname.
-b <ca_bundle_file>	Specify CA bundle file for TLS connection.
-k <privatekey_file>	Specify private key file for TLS connection
-c <certificate_file>	Specify certificate file for TLS connection
-s	Redirects all error messages to the syslog.

Related Topics

pmloadcheck

pmmasterd

pmsrvconfig

pmcheck

Syntax

```
pmcheck [ -z on|off[:<pid>] ] | [ -v ] |
          [ [ -a <string> ] [ -b ] [ -c ] [ -e <requestuser> ]
          [ -f <filename> ] [ -g <group> ] [ -h <hostname> ] [ -i ]
          [ -m <YY[YY]/MM/DD> ] [ -n <HH[:MM]> ]
          [ -o sudo|pmpolicy ] [ -p <policydir> ] [ -q  ]  [  -r <remotehost> ]
          [ -s <submithost> ] [ -t ] [ -u <runuser> ] [ command [ args ]]]
```

Description

Use the `pmcheck` command to test the policy file. Although the policy server daemon `pmmasterd` reports configuration file errors to a log file, always use `pmcheck` to verify the syntax of a policy file before you install it on a live system. You can also use the `pmcheck` command to simulate running a command to test whether a request will be accepted or rejected.

The `pmcheck` program exits with a value corresponding to the number of syntax errors found.

Options

`pmcheck` has the following options.

Table 16: Options: pmcheck

Option	Description
-a <string>	Checks if the specified string, entered during the session, matches any `alertkeysequence` configured. You can only specify this option if you supply a command. This option is only relevant when using the `pmpolicy` type.
-b	Run in batch mode. By default, `pmcheck` runs in interactive mode, and attempts to emulate the behavior of the `pmmasterd` when parsing the policy file. The `-b` option ensures that no user interaction is required if the policy file contains a password or input function; instead, a successful return code is assumed for any password authentication functions.
-c	Runs in batch mode and displays output in csv format. By default `pmcheck` runs in interactive mode. The `-c` option ensures that no user interaction is required if the policy file contains a password prompt or input function and no commands that require remote connections are attempted.

Option	Description
-e <requestuser>	Sets the value of `requestuser`. This option allows you to specify the group name to use when testing the configuration. This emulates running a session using the `sudo -u <user>` option to request that Safeguard runs the command as a particular `runuser`.
-f <filename>	Sets path to policy filename. Provides an alternative configuration filename to check. If not fully qualified, this path is interpreted as relative to the `policydir`, rather than to the current directory.
-g <group>	Sets the group name to use. If not specified, then `pmcheck` looks up the user on the master policy server host to get the group information. This option is useful for checking a user and group that does not exist on the policy server.
-h <hostname>	Specifies execution host used for testing purposes.
-i	Ignores check for `root` ownership of policy.
-m <YY [YY]/MM/DD>	Checks the policy for a particular date. Enter Date in this format: YY [YY]/MM/DD. Defaults to the current date.
-n <HH[:MM]>	Checks the policy for a particular time. Enter Time in this format: HH [:mm]. Defaults to the current time.
-o <policytype>	Interprets the policy with the specified policy type: • sudo • pmpolicy
-p policydir	Forces `pmcheck` to use a different directory to search for policy files included with a relative pathname. The default location to search for policy files is the `policydir` setting in `pm.settings`.
-q	Runs in *quiet* mode, `pmcheck` does not prompt the user for input, print any errors or prompts, or run any system commands. The exit status of `pmcheck` indicates the number of syntax errors found (0 = success). This is useful when running scripted applications that require a simple syntax check.
-r remotehost	Sets the value of the `clienthost` variable within the configuration file, useful for testing purposes. The `clienthost` variable is set to the value of the `submithost` variable.
-s submithost	Sets the value of the `submithost` variable within the configuration file, useful for testing purposes.
-t	Runs in *quiet* mode to check whether a command would be accepted or rejected. By default, `pmcheck` runs in *interactive* mode. The `-t` option ensures that no user interaction is required if the policy file contains a password prompt or input function, no output is displayed and no commands that require remote connections are attempted.

Option	Description
	Exit Status: • 0: Command accepted • 11: Password prompt encountered. The command will only be accepted if authentication is successful • 12: Command rejected • 13: Syntax error encountered
-u <runuser>	Sets the value of the `runuser` variable within the configuration file, useful for testing purposes.
-v	Displays the version number of Safeguard and exits.
-z	Enables or disables debug tracing, and optionally sends SIGHUP to running process. Refer to Enabling Program-level Tracing before using this option.
command [args]	Sets the command name and optional arguments.

You can use `pmcheck` two ways: to check the syntax of the configuration file, or to test whether a request is accepted or rejected (that is, to simulate running a command).

By default, `pmcheck` runs the configuration file interactively in the same way as `pmmasterd` and reports any syntax errors found. If you supply an argument to a command, it reports whether the requested command is accepted or rejected. You can use the `-c` and `-q` options to verify the syntax in batch or silent mode, without any user interaction required.

When you run a configuration file using `pmcheck`, you are allowed to modify the values of the incoming variables. This is useful for testing the configuration file's response to various conditions. When `pmmasterd` runs a configuration file, the incoming variables are read-only.

Example

To verify whether the sudoer policy file `/etc/sudoers`, ingoring permissions and ownership, allows user **jsmith** in the **users** group to run the `passwd root` command on host, `host1`, enter:

```
pmcheck -f /etc/sudoers -i -o sudo -u jsmith -g users
-h host1 passwd root
```

Related Topics

pmkey

pmmasterd

pmgit

Syntax

```
pmgit <subcommand> [arguments]
```

Description

The `pmgit` utility is a tool that can mediate version control operations between Subversion (SVN) and Git version control systems.

For more information on the `pmgit` policy management concept, see Managing policies in Git.

The `pmgit` utility has several subcommands, each with its own set of options and arguments. For each subcommand, `pmgit` returns `0` if the operation succeeds or `1` if an error occurred.

For more information on the subcommands, see:

- pmgit export
- pmgit Import
- pmgit Enable
- pmgit Disable
- pmgit Update
- pmgit Set
- pmgit Status
- pmgit Help

pmgit subcommands

The following topics describe `pmgit` subcommands and their arguments.

pmgit export

This subcommand exports the current policies from the SVN policy repository to a Git policy repository, and has the following arguments:

--git-url

Alias	-g
Type	string
Value	<git-policy-repository-url>
Default	N/A

Description: Sets the URL of the Git policy repository. The policy repository must be an empty bare repository, or Git will reject the operation.

Syntax

```
--git-url <git-policy-repository-url>
```

--branch

Alias	-b
Type:	string
Values:	<branch-of-policy-files>
Default:	master

Description: Sets the Git branch where the policy files are stored. If this argument is not specified, policies are exported to the default branch (master).

Syntax

```
--branch <branch-of-policy-files>
```

Example for exporting an SVN policy repository

In this example, the SVN policy repository is exported to the `exported_policies.git` Git policy repository. The URL to the Git policy repository is `https://github.com/user/exported_policies.git`, and the policies are exported to the `main` branch.

```
pmgit export --git-url https://github.com/user/exported_policies.git --
branch main
```

pmgit Import

This subcommand imports policies from an existing Git policy repository and commits the files on top of the trunk in the SVN policy repository.

--git-url

Alias	-g
Type	string
Value	<git-policy-repository-url>
Default	N/A

Description: Sets the URL of the Git policy repository. The policy repository must be an existing repository, containing the policy files.

Syntax

```
--git-url <git-policy-repository-url>
```

--branch

Alias	-b
Type:	string
Values:	<branch-of-policy-files>
Default:	master

Description: Sets the Git branch where the policy files are stored. If this argument is not provided, policies are imported from the default branch (master).

Syntax

```
--branch <branch-of-policy-files>
```

Examples

In this example, an existing Git policy repository is imported from the URL `https://github.com/user/exported_policies.git`. The branch in this example is not specified by an argument, so the policies are imported from default (master) branch.

```
pmgit import --git-url https://github.com/user/policies_to_import.git
```

pmgit Enable

This subcommand enables Git policy management. You can no longer edit policies on the policy server.

--export

Alias	-e
Type	string
Value	N/A
Default	N/A

Description: Before enabling Git policy management, policies are exported from SVN to an empty Git policy repository. This option cannot be used in conjunction with the `-m (--import)` option.

Syntax

```
--export
```

--import

Alias	-m
Type	string
Value	N/A
Default	N/A

Description: Before enabling Git policy management, policies are imported from an existing Git policy repository. This option cannot be used used in conjunction with the `-e (--export)` option.

Syntax

```
--import
```

--git-url

Alias	-g
Type	string
Value	<git-policy-repository-url>
Default	N/A

Description: Sets the URL of the Git policy repository. This option only has an effect when used in conjunction with the `-e (--export)` or `-m (--import)` options.

Syntax

```
--git-url <git-policy-repository-url>
```

--branch

Alias	-b
Type:	string
Values:	<branch-of-policy-files>
Default:	master

Description: Sets the Git branch where the policy files are stored. If this argument is not set, the default branch (master) will be used. This option only has an effect when used in conjunction with the `-e (--export)` or `-m (--import)` options.

Syntax

```
--branch <branch-of-policy-files>
```

--update-interval

Alias	-u
Type:	integer
Values:	0-60 (minutes)
Default:	5 (minutes)

Description: Sets the synchronization interval between Git and SVN. This value must be in the [0-60] minutes interval. If this argument is not set, the default value (5 minutes) will be used. If this argument is set to 0, periodic updates will be disabled. For manual update, run the `pmgit update` command.

Syntax

```
--update-interval <[0-60]>
```

--alert-script

Alias	-a
Type:	string
Values:	<path-to-script>
Default:	N/A

Description: Sets the path to a user-defined script which is run when the synchronization from Git to SVN fails due to syntax errors. This option only has an effect when used in conjunction with the `-e (--export)` or `-m (--import)` options.

Syntax

```
--alert-script <path-to-script>
```

Examples

In this example, the Git policy management is enabled, the SVN repository is exported to an empty Git policy repository, which has the URL `https://github.com/user/exported_policies.git`. The name of the branch is `main`, the update interval is set to 60 minutes, and the path to the alert script is `/path/to/script`.

```
pmgit enable
pmgit enable -e -g https://github.com/user/exported_policies.git -b main -
u 60 -a /path/to/script
```

pmgit Disable

This subcommand disables Git policy management. Policies can be managed on the policy servers with the `pmpolicy` command.

--restore

Alias	-r
Type:	string
Values:	<name-of-backup-file>
Default:	N/A

Description: When Git policy management is disabled, the SVN policy repository is restored from a previously created backup file.

Syntax

```
--restore <name-of-backup-file>
```

Examples

In this example, the Git policy management is disabled, and the repository is restored from a previously created backup file.

```
pmgit disable
pmgit disable --restore sudoers_policy_202101011000.tar
```

pmgit Update

This subcommand fetches the remote Git repository and commits the changes to the SVN policy repository.

Examples

In this example, the SVN policy repository is updated with the changes from the Git policy repository.

```
pmgit update
```

pmgit Set

This subcommand sets or updates setting in the `pm.settings` file.

--git-url

Alias	-g
Type	string
Value	<git-policy-repository-url>
Default	N/A

Description: Sets the URL of the Git policy repository. This argument can be set when the Git policy management is disabled.

Syntax

```
--git-url <git-policy-repository-url>
```

--branch

Alias	-b
Type:	string
Values:	<branch-of-policy-files>
Default:	master

Description: Sets the Git branch where the policy files are stored.

Syntax

```
--branch <branch-of-policy-files>
```

--update-interval

Alias	-u
Type:	integer
Values:	0-60 (minutes)
Default:	5 (minutes)

Description: Sets the synchronization interval between Git and SVN. This value must be in the [0-60] minutes interval. If this argument is set to 0, periodic updates will be turned off.

Syntax

```
--update-interval <[0-60]>
```

--alert-script

Alias	-a
Type:	string
Values:	<path-to-script>
Default:	N/A

Description: Sets the path to a user-defined script which is run when the synchronization from Git to SVN fails due to syntax errors.

Syntax

```
--alert-script <path-to-script>
```

Examples

In this example, the name of the branch is set to `production` and the path to the alert script is set to `/path/to/script`.

```
pmgit set -b production -a /path/to/script
```

pmgit Status

This subcommand displays information about the policy management settings.

Examples

This example shows how to display the current status of the Git policy management settings.

```
pmgit status
```

pmgit Help

This subcommand displays on-screen help. If another subcommand is given as an argument, help for that subcommand will be displayed.

Syntax

```
help <subcommand>
```

Examples

In this example, the general help section will be displayed.

```
pmgit help
```

Examples

In this example, the help for `enable` subcommand will be displayed.

```
pmgit help enable
```

`enable`

pmjoin_plugin

Syntax

```
pmjoin_plugin -h | --help [-abioqv]
                [-d <variable>=<value>] [<policy_server_host>] [-bv] -u
                [--accept] [--batch] [--define <variable>=<value>] [--
interactive]
                [--io-plugin-only][--pipestdin][--verbose] <policy_server_host>
...
                [--batch] [--verbose] -unjoin -N policy_name [--policyname
policy_name]
```

Description

Run the `pmjoin_plugin` command after installing the Sudo Plugin package (`qpm-plugin`) on a remote host to allow it to communicate with the servers in the policy group.

Options

`pmjoin_plugin` has the following options.

Table 17: Options: pmjoin_plugin

Option	Description
-a | --accept	Accepts the End User License Agreement (EULA), `/opt/quest/qpm4u/pqm4u_eula.txt`.
-b | --batch	Runss in batch mode, does not use colors or require user input.
-d <variable>=<value> | --define <variable>->=<value>	Specifies a variable for the `pm.settings` file and its associated value.
-h | --help	Displays usage information.
-i | --interactive	Runs in interactive mode, prompting for configuration parameters instead of using the default values.
-o | --io-plugin-only	Configures only the I/O logging plugin (io_plugin) without the use of the Sudo Plugin (policy_plugin) .
-q | --pipestdin	Pipes password to stdin if password is required.
-u | --unjoin	Unjoins a Sudo Plugin host from the policy server.
-N policy_name | --policyname policy_name	Use `policy_name` as the name of the policy instead of the default. This option is used to specify the name of the policy that the server should use when making policy decisions.
-v | --verbose	Displays verbose output while configuring the host.

Files

- Directory when pmjoin_plugin logs are stored: `/opt/quest/qpm4u/install`
- Sudo Plugin configuration file: `/etc/sudo.conf`

Related Topics

pmmasterd

pmsrvconfig

pmkey

Syntax

```
pmkey -v
         -a <keyfile>
         [ [-l | -r | -i <keyfile>]
         [-p <passphrase>] [-f]]
```

Description

Use the `pmkey` command to generate and install configurable certificates.

In order for a policy evaluation request to run, keys must be installed on all hosts involved in the request. The keyfile must be owned by *root* and have permissions set so only *root* can read or write the keyfile.

Options

`pmkey` has the following options.

Table 18: Options: pmkey

Option	Description
-a <keyfile>	Creates an authentication certificate.
-i <keyfile>	Installs an authentication certificate.
-l	Creates and installs a local authentication certificate to this file: `/etc/opt/quest/qpm4u/.qpm4u/.keyfiles/key_localhost` This is equivalent to running the following two commands: • `pmkey -a /etc/opt/quest/qpm4u/.qpm4u/.keyfiles/ key_localhost` -OR- • `pmkey -i /etc/opt/quest/qpm4u/.qpm4u/.keyfiles/ key_localhost`
-f	Forces the operation. For example: • Ignore the password check when installing `keyfile` using `-i` or `-r` • Overwrite existing `keyfile` when installing local `keyfile` using `-l`
-p <passphrase>	Passes the `passphrase` on the command line for the `-a` or `-l` option. If not specified, `pmkey` prompts the user for a passphrase.

Option	Description
-r	Installs all remote keys that have been copied to this directory: `/etc/opt/quest/qpm4u/.qpm4u/.keyfiles/key_<hostname>` This provides a quick way to install multiple remote keys.
-v	Displays the Safeguard version and exits.

Examples

The following command generates a new certificate, and puts it into the specified file:

```
pmkey -a <filename>
```

The following command installs the newly generated certificate from the specified file:

```
pmkey -i <filename>
```

Related Topics

pmcheck

pmmasterd

pmreplay

pmsum

pmlicense

Syntax

```
pmlicense -h
            [-c]
            -v [-c]
            -v <xmlfile> [-c]
            -l|-x <xmlfile> [-c] [-f] [-e]
            -u [s|f][-c][-d m|y][-o <outfile>][-s d|h][-t u|p|k]
            -r [e]
```

Description

The `pmlicense` command allows you to display current license information, update a license (an expired one or a temporary one before it expires) or create a new one. If you do not supply an option, then `pmlicense` displays a summary of the combined licenses configured on this host.

Options

`pmlicense` has the following options.

Table 19: Options: pmlicense

Option	Description
-c	Displays output in CSV, rather than human-readable format.
-d	Filters a license report; restricting the date to either: • **m**: Only report licenses used in the past month. • **y**: Only report licenses used in the past year.
-e	Does not forward the license change to the other servers in the group.
-f	Does not prompt for confirmation in interactive mode.
-h	Displays usage.
-l <xmlfile>	Configures the selected XML license file, and forwards it to the other servers in the policy group. This option must be run as the root user or a member of the pmpolicy group.
-o <outfile>	Sends report output to selected file rather than to the default. For csv output, the default is file: `/tmp/pmlicense_report_<uid>.txt`; for human-readable output, the default is stdout.
-r	Regenerates and configures the default trial license, removing any configured commercial licenses, and forwards this change to the other servers in the policy group.
-s	Sort the report data by either: • **d**: date (newest first) • **h**: hostname (lowest first)
-t	Filters license report by client type: • **u**: Privilege Manager for Unix client • **p**: sudo policy plugin • **k**: sudo keystroke plugin

Option	Description
-u	Displays the current license utilization on the master policy server: • **s**: Show summary of hosts licensed • **f**: Show full details of hosts licensed, with last used times
-v	If you do not provide a file argument, it displays the details of the currently configured license. If you provide a file argument, it verifies the selected XML license file and displays the license details.
-x <xmlfile>	Configures the selected XML license file. This option is deprecated, use the "-l" option instead.

License data is updated periodically by the `pmloadcheck` daemon. See pmloadcheck on page 147 for details.

Examples

To display current license status information, enter the following:

```
# pmlicense
```

Safeguard displays the current license information, noting the status of the license. The output will be similar to the following:

```
*** One Identity Safeguard ***
*** Privilege Manager for Unix VERSION 6.n.n (nnn) ***
*** CHECKING LICENSE ON HOSTNAME:<host>, IP address: <IP>
*** SUMMARY OF ALL LICENSES CURRENTLY INSTALLED ***
*License Type                                  PERMANENT
*Commercial/Freeware License                   COMMERCIAL
*Expiration Date                               NEVER
*Max QPM4U Client Licenses                     1000
*Max Sudo Policy Plugin Licenses               0
*Max Sudo Keystroke Plugin Licenses            0
*Authorization Policy Type permitted           ALL
*Total QPM4U Client Licenses In Use            2
*Total Sudo Policy Plugins Licenses in Use     0
*Total Sudo Keystroke Plugins Licenses in Use  0

*** LICENSE DETAILS FOR PRODUCT:QPM4U
*License Version                               1.0
*Licensed to company                           Testing
*Licensed Product                              QPM4U(1)
```

```
*License Type                                   PERMANENT
*Commercial/Freeware License                    COMMERCIAL
*License Status                                 VALID
*License Key                                    PSXG-GPRH-PIGF-QDYV
*License tied to IP Address                     NO
*License Creation Date                          Tue Feb 08 2012
*Expiration Date                                NEVER
*Number of Hosts                                1000
```

To update or create a new a license, enter the following at the command line:

```
pmlicense -l <xmldoc>
```

Safeguard displays the current license information, noting the status of the license, and then validates the information in the selected `.xml` file, for example:

```
*** One Identity Safeguard for Sudo ***
*** Safeguard for Sudo VERSION 7.n.n (nnn) ***
*** CHECKING LICENSE ON HOSTNAME:<host>, IP address:<IP> ***
*** SUMMARY OF ALL LICENSES CURRENTLY INSTALLED ***
*License Type                                   PERMANENT
*Commercial/Freeware License                    COMMERCIAL
*Expiration Date                                NEVER
*Max QPM4U Client Licenses                      1000
*Max Sudo Policy Plugin Licenses                0
*Max Sudo Keystroke Plugin Licenses             0
*Authorization Policy Type permitted            ALL
*Total QPM4U Client Licenses In Use             2
*Total Sudo Policy Plugins Licenses in Use      0
*Total Sudo Keystroke Plugins Licenses in Use   0
*** Validating license file: <xmldoc> ***
*** LICENSE DETAILS FOR PRODUCT:QPM4U
*License Version                                1.0
*Licensed to company                            Testing
*Licensed Product                               QPM4U(1)
*License Type                                   PERMANENT
*Commercial/Freeware License                    COMMERCIAL
*License Status                                 VALID
*License Key                                    PNFT-FDIO-YSLX-JBBH
*License tied to IP Address                     NO
*License Creation Date                          Tue Feb 08 2012
*Expiration Date                                NEVER
*Number of Hosts                                100
*** The selected license file (<xmldoc>) contains a valid license ***
```

```
Would you like to install the new license? y
Type y to update the current license.
Archiving current license… [OK]
*** Successfully installed new license ***
```

Related Topics

pmmasterd

Installing licenses

Displaying license usage

pmloadcheck

Syntax

```
pmloadcheck -v
               -s|-p|-i [-e <interval>][-t <sec>]
               [-c|-f][-b][ -h <master>][-t <sec>] [-a][-r]
```

Description

The pmloadcheck daemon runs on Safeguard for Sudo policy servers. By default, every 60 minutes the daemon verifies the status of the configured policy servers. It controls load balancing and failover for connections made from the host to the configured policy servers, and on secondary servers, it sends license data to the primary server.

When the `pmloadcheck` daemon runs, it attempts to establish a connection with the policy servers to determine their current status. If `pmloadcheck` successfully establishes a session with a policy server, it is marked as *online* and is made available for normal client sessions. If `pmloadcheck` does not successfully establish a session with a policy server, it is marked as *offline*.

Information is gathered from a policy server each time a normal client session connects to the policy server. This information is used to determine which policy server to use the next time a session is requested. If an agent cannot establish a connection to a policy server because, for example, the policy server is offline, then this policy server is marked as *offline* and no more connections are submitted to this policy server until it is marked available again.

To check the current status of all configured policy servers, and display a brief summary of their status, run `pmloadcheck` with no options. Add the `-f` option to show full details of each policy server status.

Options

`pmloadcheck` has the following options.

Table 20: Options: pmloadcheck

Option	Description
-a	Verifies the connection as if certificates were configured.
-b	Runs in batch mode.
-c	Displays output in CSV format.
-e <interval>	Sets the refresh interval (in minutes) to determine how often the `pmloadcheck` daemon checks the policy server status. Default = 60.
-f	Shows full details of the policy server status when verifying and displaying policy server status.
-h <master>	Selects a policy server to verify.
-i	Starts up the `pmloadcheck` daemon, or prompt an immediate recheck of the policy server status if it is already running.
-P	Sends SIGNUP to a running daemon.
-p	Pauses (sends SIGUSR1) to a running daemon.
-r	Reports last cached data for selected servers instead of connecting to each one.
-s	Stops the `pmloadcheck` daemon if it is running.
-t <sec>	Specifies a timeout (in seconds) to use for each connection.
-v	Displays the version string and exits.

pmlog

Syntax

```
pmlog [-dlvq] [-p|a|e|r|x <printexpr>] [-f <filename>] [[-c] <constraint>]
         [[-c] <constraint>] [-f <filename>] -h
         [--user <username>]
         [--runuser <username>] [--runhost <hostname>] [--reqhost <hostname>]
         [--masterhost <hostname>][--command <pattern>] [--reqcommand <pattern>]
```

```
[--runcommand <pattern>][--before "<YYYY/MM/DD hh:mm:ss>"]
[--after "<YYYY/MM/DD hh:mm:ss>"][--result Accept|Reject]
```

Description

Use the `pmlog` command to selectively choose and display entries in a Privilege Manager for Unix event log. Each time a job is accepted, rejected, or completed by `pmmasterd`, an entry is appended to the file specified by the `eventlog` variable in the configuration file. `eventlog` is sent to `/var/opt/quest/qpm4u/pmevents.db` on all platforms.

Options

`pmlog` has the following options.

Table 21: Options: pmlog

Option	Description
-a <expression>	Sets the print expression for accept events to the specified expression.
-c <constraint>	Selects particular entries to print; specify **constraint** as a Boolean expression. See Examples.
-d	Dumps each entry as it is read without matching 'accept' and 'end' entries. The `-d` (dump) option forces `pmlog` to print each entry as it is read from the file. The default output format includes a unique identifier at the start of each record, allowing 'end' events to be matched with 'accept' events.
-e <expression>	Sets the print expression for finish events to the specified expression.
-f <filename>	Reads the event log information from the specified file.
-h	Displays usage information.
-l	Dumps alert log entries only.
-p <expression>	Sets the print expression for all event types to the specified expression.
-q	Runs in quiet mode; no expression errors (for example, undefined variables) are printed.
-r <expression>	Sets the print expression for reject events to the specified expression.
-v	Turns on verbose mode.

Option	Description
-x <expression>	Sets the print expression for alert events to the specified expression.
Quick Search Options	
--user <username>	Selects entries in which the requesting user matches `username`.
--runuser <username>	Selects entries in which `runuser` matches `username`.
--runhost <hostname>	Selects entries in which `runhost` matches `hostname`.
--reqhost <hostname>	Selects entries in which the requesting host matches `hostname`.
--masterhost <hostname>	Selects entries in which `masterhost` matches `hostname`.
--command <pattern>	Selects entries in which the requested command matches `pattern`.
--reqcommand <pattern>	Return events where the given text appears anywhere in the requested command line.
--runcommand <pattern>	Selects entries in which the `runcommand` host matches `pattern`.
--before "<YYYY/MM/DD hh:mm:ss>"	Selects entries occurring before the specified date and time.
--after "<YYYY/MM/DD hh:mm:ss>"	Selects entries occurring after the specified date and time.
--result Accept|Reject	Selects entries that were accepted or rejected.

Examples

Without arguments, `pmlog` reads the default `eventlog` file and prints all its entries. If you have chosen a different location for the event log, use the `-f` option to specify the file for `pmlog`.

By default, `pmlog` displays one entry for each completed session (either rejected or accepted). You can filter the results to print only entries which satisfy the specified

constraint using the -c option. In these examples the -c option is used to specify a constraint as a Boolean expression:

```
pmlog -c'event=="Reject"'
```

```
pmlog -c'date > "2008/02/11"'
```

```
pmlog -c'user=="dan"'
```

which prints only rejected entries, entries that occur after February 11, 2008, or requests by user Dan, respectively.

See Safeguard Variables on page 76 for more information about policy variables.

The following options accept shortcut notations to specify constraints:

- --user username
- --runuser username
- --reqhost hostname
- --runhost hostname
- --masterhost hostname
- --command command
- --runcommand command
- --reqcommand command
- --before "YYYY/MM/DD hh:mm:ss"
- --after "YYYY/MM/DD hh:mm:ss"
- --result Accept|Reject

For example, here are equivalent constraints to the previous example specified using shortcuts:

```
pmlog --result Reject
```

```
pmlog --after "2008/02/11 00:00:00"
```

```
pmlog --user dan
```

With shortcuts, you can express user names and host names as patterns containing wild card characters (? and *). For example, to display entries for all requests for user1, user2, and user3, use the following shortcut:

```
pmlog --user “user?"
```

Enclose patterns containing wild card characters in quotes to avoid being interpreted by the command shell.

Use the `-d` and `-v` options for debugging. Normally, when `pmlog` finds an 'accept' entry, it refrains from printing until the matching 'end' entry is found; all requested information including `exitstatus`, `exitdate`, and `exittime` is then available to print.

The `-d` (dump) option forces `pmlog` to print each entry as it is read from the file. The default output format includes a unique identifier at the start of each record, allowing 'end' events to be matched with 'accept' events.

The `-v` (verbose) option prints all the variables stored with each entry.

The `-t` option turns on *tail follow* mode. The program enters an endless loop, sleeping and printing new event records as they are appended to the end of the log file. The `-d` flag is implied when using `-t`.

You can specify the output format for each of the three event types - 'accept', 'reject' or 'finish' - with the `-a`, `-r`, and `-e` options. Use the `-p` option to set the output for all three event types.

For example, to print only the dates and names of people making requests, enter:

```
pmlog -p'date + "\t" + user + "\t" + event'
```

-OR-

```
pmlog -p 'sprintf("%s %-8s %s", date, user, event)'
```

See Listing event logs on page 60 for more examples of using the `pmlog` command.

Note that if you run `pmlog --csv console` to obtain CSV output from pmlog, refer to pmlogsearch on page 156 for a list of the column headings.

pmlogadm

Syntax

```
pmlogadmin> archive <event_log_path> <archive_path> --before <YYYY-MM-DD>
           [--clean-source] [--dest-dir <destination_path>] [--no-zip]
pmlogadmin> archive <event_log_path> <archive_path> --older-than <days>
           [--clean-source] [--dest-dir <destination_path>] [--no-zip]
pmlogadmin> backup <event_log_path> <backup_path>
```

```
pmlogadmin> create <new_event_log_path>
pmlogadmin> encrypt enable|disable|rekey <event_log_path>
pmlogadmin> help [<command>]
pmlogadmin> import [-y|-n] <source_event_log> <dest_event_log>
pmlogadmin> info <event_log_path>
pmlogadmin> --help|-h
pmlogadmin> --version|-v
```

Description

Privilege Manager event log administration utility. Use `pmlogadm` to manage encryption options on the event log.

Options

`pmlogadm` has the following options.

Table 22: Options: pmlogadm

Option	Description
-h, --help	Displays usage information. `help [<command>]` By default the `help` command displays the general usage output. When you specify a command, it displays a usage summary for that command.
-v, --version	Displays the version number of Safeguard and exits.

Table 23: Global options: pmlogadm

Option	Description
--verbose	Enables verbose output.
--silent	Disables all output to stdout. Errors are output to stderr.

Table 24: Valid commands: pmlogadm

Option	Description

archive	Moves old events to an archive.
	`archive <event_log_path> <archive_name> --before <YYYY-MM-DD> [--cleansource] [--dest-dir <destination_path>] [--no-zip]`
	-OR-
	`archive <event_log_path> <archive_name> --older-than <days> [--cleansource] [--dest-dir <destination_path>] [--no-zip]`

Option	Description
	Moves events that occurred before the indicated date (YYYY-MM-DD) to an archive-named `archive_name`. If you use the second form, specify the date as days before the current date. The archive is created in the current working directory unless you specify a destination path using the `--dest-dir` option. By default, the archive is compressed using `tar` and `gzip`, but you can skip this using the `--no-zip` option, in which case the resulting archive is a directory containing the new log with the archived events. All files in that directory are required to access the archive. To access the archive, use `pmlog`. Moving events to an archive may not reduce the actual file size of the event log. To reduce the file size, the source log must be cleaned. To clean the source log, add the `--clean-source` option. When a large number of events are present in the source log this option can increase the archive process time and use a large amount of disk space while the process runs. Once started, do not interrupt the process.
backup	Creates a backup of the source log (`event_log_path`), in location `backup_log`.
create	Creates new empty audit files for that log. `create <new_event_log_path>` This may include a keyfile which has the `-kf` suffix, a journal file with the `-wal` suffix, and a `-shm` system file. It is critical that the group of files that make up an event log remain together at all times. Removal of any one of these files may result in permanent loss of access to the event log.
encrypt	Enables or disables encryption of an event log. `encrypt enable|disable|rekey <event_log_path>` By default all event logs created by Safeguard are encrypted using the AES-256 standard. The encryption key is stored in the keyfile which is in the same path as the event log and has the same file name, and the `-kf` suffix. It is critical that this file remain in the same path as the main event log file. You can decrypt the whole log file using the encrypt disable command, passing the path of the main event log file as an argument. Enable encryption using `encrypt enable`. The `encrypt rekey` command generates a new encryption key and re-encrypt all data in the event log using that new key data. The key file is automatically updated with the new key data if the operation succeeds.
import	Imports events. `import [-y|-n] <source_event_log> <dest_event_log>`

Option	Description
	Import events from `source_event_log`, adding them to `dest_event_log`.
info	Displays information about the event log. `info <event_log_path>` Displays information about the `event_log_path`. The information reported includes the current encryption status of the event log, the size of the file and the number of events contained in the log.

Settings

The following entries in the `/etc/opt/quest/qpm4u/pm.settings` file are used by `pmlogadm`

Table 25: Settings: pmlogadm

Option	Description
eventLogQueue <pathname>	Specify the location of the event log queue, used by both `pmmasterd` and `pmlogsrvd`. This option is only used to determine whether the `pmlogsrvd` service is currently running.

For more usage information for a specific command, run:

`pmlogadm help <command>`

Files

The default Privilege Manager event log file is located at:

`/var/opt/quest/qpm4u/pmevents.db`

Other files that may be used by `pmlogadm` are:

- settings file: `/etc/opt/quest/qpm4u/pm.settings`
- pid file: `/var/opt/quest/qpm4u/evcache/pmlogsrvd.pid`

Related Topics

pmlog

pmlogsrvd

pmmasterd

pmlogsearch

Syntax

```
pmlogsearch [--csv] [--no-sort]
               [--before "<YYYY/MM/DD hh:mm:ss>"] [--after "<YYYY/MM/DD
hh:mm:ss>"]
               [--user <username>] [--host <hostname>] [--result accept|reject]
               [--text <keyword>]
               -h | --help
               -v | --version
```

Description

Use the `pmlogsearch` command to perform a search on all logs in this policy group based on specified criteria.

You must specify at least one search condition; you can combine conditions.

Options

`pmlogsearch` has the following options.

Table 26: Options: pmlogsearch

Option	Description
--csv	Outputs the search results in CSV format, suitable for consumption by Safeguard. If this option is not present, the output is human-readable. One or more of the search criteria must be present, and any combination of the criteria is accepted. When multiple criteria are present they must all be matched (that is, the query criteria are combined using AND logic) for a log to be included in the results.

--after
--before
	Returns logs generated for sessions initiated after or before the specified time and date. For example:
	`# pmlogsearch --after “2012/01/04 00:00:00”`
	returns all logs for sessions since January 4, 2012.
	`# pmlogsearch --after “2012/01/01 00:00:00” --before “2012/12/31 23:59”`
	returns all logs generated during 2012.

--user <username>	Searches for logs generated by sessions requested by the specified user name. `username` is case sensitive. For example: `# pmlogsearch --user harry`

Option	Description
	returns the locations of all keystrokelogs for sessions requested by the user named "harry". The pattern may include the following wild card symbols: • * = match any string • ? = match any single character

--host <hostname>	Searches for logs generated by sessions that ran on hosts matching the given pattern. The pattern may include the following wild card symbols:
	• * = match any string
	• ? = match any single character
	For example:
	`# pmlogsearch --host “myhost?.mydomain.com”`
	matches logs for sessions that ran on `myhost1.mydomain.com` or `myhost2.mydomain.com`, but not `myhost1` or `myhost10.mydomain.com`.
	`# pmlogsearch --host “myhost*”`
	matches logs for sessions that ran on `myhost1.mydomain.com`, `myhost2.mydomain.com`, `myhost1` or `myhost10.mydomain.com`, but will not match `anotherhost.mydomain.com`.
	`# pmlogsearch --host myhost11.mydomain.com`
	only matches logs for sessions that ran on host `myhost11.mydomain.com`.

--result	Returns only events with the indicated result.

--text “<keyword>”	Searches for events where the specified text occurs in the command line or events with keystroke logs that contain the specified text.
	You must enter the keyword or phrase as one argument. If the phrase contains a space, enclose the whole phrase in quotes. For example:
	`# pmlogsearch --text “my phrase”`
	matches any log containing the string "my phrase".
	`# pmlogsearch --text phone`
	matches logs containing any word with the substring `phone` (such as, telephone, headphones, phones), or the complete word phone.

--no-sort	Does not sort the results.
-v | --version	Displays the version number of Safeguard and exits.
-h | --help	Displays usage information and exits.

Output

You can output the search results in either human-readable or CSV format.

Human-Readable Output

The following is an example of the human-readable output of a search:

```
# pmlogsearch --user sheldon --text Linux
Search matches 5 events
2012/01/19 18:12:25 : Accept : sheldon@host1.example.com
   Request: sheldon@host1.example.com : uname -a
Executed: root@host1.example.com : uname -a
   IO Log: pmsrv1.example.com: opt/quest/qpm4u/iologs/sheldon/root/uname-
20120119-181225.OiaiBr
2012/01/19 18:11:56 : Accept : sheldon@host1.example.com
   Request: sheldon@host1.example.com : uname -a
Executed: root@host1.example.com : uname -a
   IO Log: pmsrv2.example.com: opt/quest/qpm4u/iologs/sheldon/root/uname-
20120119-181156.x46qJP
2012/01/19 17:59:09 : Accept : sheldon@host2.example.com
   Request: sheldon@host2.example.com : uname -a
Executed: root@host2.example.com : uname -a
   IO Log: pmsrv2.example.com: opt/quest/qpm4u/iologs/sheldon/root/uname-
20120119-175909.1H0P5n
2012/01/19 17:58:42 : Accept : sheldon@host2.example.com
   Request: sheldon@host2.example.com : uname -a
Executed: root@host2.example.com : uname -a
   IO Log: pmsrv2.example.com: opt/quest/qpm4u/iologs/sheldon/root/uname-
20120119-175842.ZvfrMv
2012/01/19 17:58:14 : Accept : sheldon@host2.example.com
   Request: sheldon@host2.example.com : uname -a
Executed: root@host2.example.com : uname -a
   IO Log: pmsrv1.example.com: opt/quest/qpm4u/iologs/sheldon/root/uname-
20120119-175814.
```

CVS output

The results are output in CSV format, without field headings. The columns are listed in order below:

1. Session date/time
2. Session Unique ID
3. Master host
4. Submit host (host from which the session was requested)
5. Submit user (the user that requested the session)
6. Requested host
7. Requested user account
8. Requested command line
9. Result (Accept/Reject)

10. Run host (the host on which the command was run)
11. Run user (the user account used to run the command)
12. Command line that ran
13. The exit return code if the command ran successfully, or "NO_EXIT" if the event was rejected or the command failed to run
14. Keystroke log host. This column is blank, if it is the same as #3 Master host.
15. Keystroke log file path

The following is an example of CSV output:

```
# pmlogsearch --csv --user penny --text "Linux"
"2012/01/19 18:10:40", "4d3729207eec", "pmsrv1.example.com",
"host1.example.com", "penny", "uname", "Accept", "host1.example.com", "penny",
"uname", "pmsrv1.example.com",
"opt/quest/qpm4u/iologs/host1.example.com/penny/uname-20120119-181040.hLqZFY"
"2012/01/19 18:10:13", "4d3729057e5f", "pmsrv1.example.com",
"host1.example.com", "penny", "uname", "Accept", "host1.example.com", "penny",
"uname", "pmsrv1.example.com",
"opt/quest/qpm4u/iologs/host1.example.com/penny/uname-20120119-181013.yG1m41"
"2012/01/19 18:00:14", "4d3726ae1ec0", "pmsrv2.example.com",
"host1.example.com", "penny", "uname", "Accept", "host1.example.com", "penny",
"uname", "pmsrv2.example.com",
"opt/quest/qpm4u/iologs/host1.example.com/penny/uname-20120119-180015.Z42heZ"
"2012/01/19 18:00:47", "4d3726cf1f9d", "pmsrv1.example.com",
"host1.example.com", "penny", "uname", "Accept", "host1.example.com", "penny",
"uname", "pmsrv1.example.com",
"opt/quest/qpm4u/iologs/host1.example.com/penny/uname-20120119-180047.GUtrRt"
```

Related Topics

Viewing the log files using command line tools

pmlogsrvd

Syntax

```
pmlogsrvd [-d | --debug] [-h | --help] [--log-level <level>] [--no-detach]
            [--once] [-q | --queue <queue_path>] [--syslog [facility]]
            [-t | --timeout <delay_seconds>] [-v | --version]
```

Description

`pmlogsrvd` is the Safeguard log access daemon, the service responsible for committing events to the Safeguard event log, and managing the database storage used by the event log.

When an incoming event is processed by `pmmasterd` that event must be logged to the event log. `pmmasterd` commits a record of the log to the event log queue, which is monitored by `pmlogsrvd`. `pmlogsrvd` takes each event from the queue and commits that event to the actual event log.

Options

`pmlogsrvd` has the following options.

Table 27: Options: pmlogsrvd

Option	Description
-d | --debug	Enables debug operation. This option prevents `pmlogsrvd` from running in the background, and enables debug output to both the log and the terminal.
-h | --help	Displays the usage information and exits.
--log-level <level>	Controls the level of log messages included in the log file. By default the logging level logs only error messages. Valid logging levels, in ascending order by volume of messages, are: • none • error • warning • info • debug
--no-detach	Do not run in the background or create a pid file. By default, `pmlogsrvd` forks and runs as a background daemon. When you specify the `--no-detach` option, it stays in the foreground.
--once	Processes the queue once immediately and then exits.
-q | --queue <path>	Specifies the location of the event log queue as **path**.
--syslog	Enables logging to syslog.
-t | --timeout <delay_ seconds>	Specifies the time delay between processing the queue as time seconds. By default `pmlogsrvd` waits for 120 seconds before waking to scan the event log queue if no other trigger causes it to begin processing. Normally processing is triggered directly by `pmmasterd` immediately after an event is processed.
-v | --version	Displays the version number of Safeguard and exits.

Settings

`pmlogsrvd` uses the following entries in the `/etc/opt/quest/qpm4u/pm.settings` file.

Table 28: Settings: pmlogsrvd

Setting	Description
eventLogQueue <pathname>	Specifies the location of the event log queue, used by both `pmmasterd` and `pmlogsrvd`. This setting is ignored by `pmlogsrvd` when you use the `--queue` option on the command line.
pmlogsrvlog <pathname>	Fully qualified path to the `pmlogsrvd` log file.
syslog yes|no	By default, `/pmlogsrvd/fR` used this setting to determine whether to send log messages to syslog. When you use the `/syslog/fR` option on the command line, this setting is ignored.

Files

- settings file: `/etc/opt/quest/qpm4u/pm.settings`
- pid file: `/var/opt/quest/qpm4u/evcache/pmlogsrvd.pid`

Related Topics

pmlog

pmlogsearch

pmmasterd

pmlogxfer

Syntax

```
pmlogxfer -h | -v
```

Description

Transfers event logs and I/O logs after an off-line policy evaluation has occurred. `pmlogxfer` is initiated by `pmloadcheck` when there are log files queued for transfer from a Sudo Plugin host to the server.

Note that `pmlogxfer` is not intended to be run directly, it is normally invoked by `pmpluginloadcheck` at a regular interval (every 30 minutes by default).

Options

`pmlogxfer` has the following options.

Table 29: Options: pmlogxfer

Option	Description
-h	Displays usage information.
-v	Displays the version number of Safeguard and exits.

Files

Directory for offline log files:

```
/var/opt/quest/qpm4u/offline
```

Related Topics

pmpluginloadcheck

pmmasterd

Syntax

```
pmmasterd [ -v ]| [ [ -ars ] [ -e <logfile> ] ]
```

Description

The Safeguard master daemon (`pmmasterd`) is the policy server decision-maker. `pmmasterd` receives requests from `pmrun` or the Sudo Plugin and evaluates them according to the security policy. If the request is accepted, `pmmasterd` asks `pmlocald` or the Sudo Plugin to run the request in a controlled account such as `root`.

A connection is maintained between `pmmasterd` and the Sudo Plugin for the duration of the session. This also occurs between `pmmasterd` and `pmlocald`, if keystroke logging is enabled. When the `pmmasterd` connection is maintained throughout the session, keystroke and event log data is forwarded on this connection.

If keystroke logging is not enabled, `pmlocald` reconnects to `pmmasterd` at the end of the session to write the event log record showing the final completion code for the command run by `pmlocald`. If `pmlocald` is unable to reconnect, it writes instead to a holding file, `pm.eventhold.hostname`. It then attempts to write the `pmevents.db` record to the host the next time `pmmasterd` connects to `pmlocald`. Multiple files can accrue and they will all be delivered to the proper host when the connection is restored.

The policy server master daemon typically resides on a secure machine. You can have more than one policy server master daemon on different hosts for redundancy or to serve multiple networks.

`pmmasterd` logs all errors in a log file if you specify the `-e filename` option.

Options

`pmmasterd` has the following options.

Table 30: Options: pmmasterd

Option	Description
-a	Sends job acceptance messages to syslog.
-e <filename>	Logs any policy server master daemon errors in the file specified.
-r	Sends job rejection messages to syslog.
-s	Sends any policy server master daemon errors to `syslog`.
-v	Displays the version number of `pmmasterd` and exits.

Files

- Safeguard policy file (sudo type): `/etc/opt/quest/qpm4u/policy/sudoers`

Related Topics

pmcheck

pmkey

pmreplay

Safeguard for Sudo Policy Evaluation

pmpluginfo

Syntax

```
pmpluginfo -v | -c [-h <host>]
```

Description

Run the `pmpluginfo` command on a Sudo Plugin host to display information about the policy server group that the host has joined.

Options

`pmplugininfo` has the following options.

Table 31: Options: pmplugininfo

Option	Description
-c	Displays output in CSV, rather than human-readable format.
-h <hostname>	Specifies the hostname to interrogate for policy group information.
-v	Displays product version and exits.

Examples

The following is an example of the human-readable output:

```
Joined to a policy group             : YES
Name of policy group                 : adminGroup1
Hostname of primary policy server    : adminhost1
```

Related Topics

Checking the Sudo Plugin configuration status

Sudo policy is not working properly

pmpluginloadcheck

Syntax

```
pmpluginloadcheck -v
                    -s|-p|-i [-e <interval>][-t <sec>]
                  [-c|-f][-b][ -h <master>][-t <sec>] [-a][-r]
```

Description

The `pmpluginloadcheck` daemon runs on each Sudo Plugin host and controls load balancing and failover for connections made from the host to the configured policy servers. It runs as a daemon, and is started as needed to verify the status of the configured policy servers.

Information is gathered from a policy server each time a normal sudo session connects to the policy server. This information is used to determine which policy server to use the next time a session is requested. If a host cannot establish a connection to a policy server

because, for example, the policy server is offline, then this policy server is marked as offline and no more connections are submitted to this policy server until it is available again. For each policy server that is marked as offline, the `pmpluginloadcheck` daemon checks at intervals, and attempts to establish a connection with the policy server to determine its current status. If `pmpluginloadcheck` successfully establishes a session with the policy server, it is marked as online and is made available for normal sudo sessions.

To check the current status of all configured policy servers and display a brief summary of their status, run `pmpluginloadcheck` with no options. Add the `-f` option to show full details of each policy server status.

Options

`pmpluginloadcheck` has the following options.

Table 32: Options: pmpluginloadcheck

Option	Description
-a	Verifies the connection as if certificates are configured.
-b	Runs in batch mode.
-c	Reports full details of selected servers in CSV, rather than human-readable format.
-e <interval>	Sets the refresh interval (in minutes). The default is 60 minutes. The minimum value is 2 minutes.
-f	Reports full details of data for each policy server (or selected policy server, when using the `-h` option).
-h <master>	Selects a policy server to verify.
-i	Starts up the `pmpluginloadcheck` daemon, if it is not already running.
-P	Pause (send SIGUSR1) to a running daemon.
-p	Sends SIGHUP to a running daemon.
-r	Reports last cached data for selected servers instead of connecting.
-s	Stops the `pmloadcheck` daemon, if it is running.
-t <sec>	Specifies a timeout (in seconds) to use for each connection.
-v	Displays the version string and exits.

pmpolicy

Syntax

```
pmpolicy -v  command [args] [-c] [<command>.] -h
```

Description

`pmpolicy` is a command line utility for managing the Privilege Manager for Unix security policy. Use the `pmpolicy` command to view and edit the policy in use by the group. Any user in the `pmpolicy` group may run this command on any configured policy server host.

This utility checks out the current version, checks in an updated version, and reports on the repository.

You can use the `-c` option to display the result of the command in CSV, rather than in a human-readable form. The CVS output displays the following fields: Resultcode, name, description, Output msg.

The `pmpolicy` utility exits with the following possible exit status codes, unless otherwise stated below:

Exit status codes

- 0: Success
- 1: Repository does not exist
- 2: Specified path does not exist
- 3: Failed to checkout from the repository
- 4: Failed to check in to the repository
- 5: Syntax error found in new policy – check in was abandoned
- 6: Conflict found when attempting a check in - check in was abandoned
- 7: Policy type not found in repository
- 8: Failed to access the repository to report requested information
- 9: The selected version was not found in the repository
- 10: Directory did not contain a working copy
- 11: Check in abandoned
- 12: Invalid path specified
- 13: Invalid configuration

Options

The following is a summary of the commands and options available to `pmpolicy`.

Run any command with a `-h` to get more information about it. For example:

```
pmpolicy <command> -h
```

Table 33: Commands and options: pmpolicy

Command	Description
add	Adds a new file from the specified path to the policy repository. `add -p path -d dir [-n [-l commitmsg]] [-c] [-u <user>]` Records the addition of a new file to the working copy of the policy. Use the `-p` option to specify the file path (relative to the top-level directory in the policy) to add. Use the `-d` option to specify the directory of the working copy. The `-n` option commits the changes to the repository. If you use the `-n` option, you can also use the `-l` option to provide a commit log message. If you use `-n` without the `-l`, the command interactively prompts you for the commit log message

checkout	Checks out a working copy of the policy to the specified directory.
	`checkout -d <dir> [-c] [-r <revision>]`
	If the directory does not exist, it is created. If the selected directory exists, the existing contents is overwritten. By default, the latest copy is retrieved; use the `-r` option to check out a particular revision. You can specify a revision using SVN DATE format, or the HEAD keyword, as well as revision numbers.
	A date format specified without a time, defaults to 00:00:00.
	The earliest time you can use to identify a particular revision is one second after the time you commit the revision. For example, if you committed revision 2 at 12:00:00, then you must specify a time of 12:00:01 or later to check out revision 2. For example:
	`pmpolicy checkout -d /tmp -r "{2012-01-02 12:00:01}" # checkout revision that existed on 2012-01-02 00:00:00`

commit	Checks in changes from a working copy to the policy repository. `commit -d <dir> [-l <commitmsg>] [-c] [-a force|-abort|merge|overwrite][-u <user>]` Commits the working copy of the policy from the indicated directory. All files in the indicated directory are checked in to the repository. This working copy is first verified for syntax errors using the `pmcheck` utility. The working copy must match the policy type currently in use, otherwise a syntax error will be produced by `pmcheck`.

Command	Description
	If no syntax errors are encountered, it attempts to check in this copy into the repository, honoring the `-a` option as described below. Exit status of 0 indicates successful check in. The `-a` option indicates the action to be taken when checking in a working copy, if the repository has changed since the working copy was checked out, that is, the edits are based on an out-of-date copy of the repository. The resulting differences between the working copy and the repository may or may not conflict. You can specify the following actions: • **Merge**: If the only differences are non-conflicting, then merge the changes. If any conflicting changes are found, abort the check in. • **Overwrite**: Merge the changes. If any conflicting changes are found in the repository, select those from the working copy. • **Force**: Overwrite the copy in the repository with the working copy, discarding any changes that have been committed since the working copy was checked out. • **Abort**: Abandon the check in if the working copy is out of date, regardless of whether changes are in conflict (this is the default) For example: `pmpolicy commit -d /tmp -a force`
diff	Checks the differences between two revisions of the policy and reports the output to `stdout`, or to the selected output file. `diff [-o <outfile>][-c][-f][-p <path>][-d <dir> [-r <v1>]] | [-r [<v1>:[<v2>]]` By default, this option displays the differences between the two selected revisions. If you specify the `-f` option, it displays the incremental differences between each revision in the specified range. You can specify revisions using any acceptable SVN revision format, such as HEAD, COMMITTED, or DATE format. You can use the `-o` option to report the "diff" output to a file, rather than to `stdout` (the default). • If you specify a directory, it compares the copy in that directory with the selected revision (or the latest revision in the repository, if you do not specify a revision). • If you specify one revision, it reports the difference between the latest and selected revision. • If you specify two revisions, it reports the difference between the selected revisions.

Command	Description
	Exit status codes: • 0: no differences were detected. • 1: differences were detected • 2: An error occurred For example: `pmpolicy diff -d /tmp -o /tmp/diffs.txt -r2 pmpolicy diff -r1:2 -o /tmp/diffs.txt`
edit	The utility checks out a temporary working copy of the policy and starts the appropriate interactive editor to edit the files. For a sudo policy, it runs `visudo`; for a legacy policy it uses `$EDITOR`. `edit [-a force|abort|merge|overwrite] [-l <commitmsg>] [-p <path>][-u <user>]` This option is useful for manual interactive editing of the policy on the command line. On completion of the edit, it verifies the syntax of the policy. If no errors are found, it checks the edits back in to the repository. If any errors are found, then it exits without checking in the changes. When saving an edited policy, some non-ASCII characters in the commit log message may error and cause all changes to the policy to be discarded. To avoid this possibility, avoid using backspace, arrow keys and any other keys that may be interpreted as non-ASCII characters within the shell.
help	Displays usage information.
log	Logs revision information about the repository. `log [-o <outfile>][-c][-e][-r <revision>]` Reports information about the repository to `stdout` or to the selected output file. This displays details of the user who changed the repository, the version number for this change, along with the time and date of the change. By default, this option shows details of each revision in the repository, one version per line. If you specify a version, it shows the details of this version. You can use the `-o` option to report the "log" output to a file, rather than to `stdout`. The status is displayed in the following format for CSV output:

Command	Description

	`"<version>","<username>",<YYYY-MM-DD>,<HH:MM:SS>"<commitmsg>"`
	For example:
	`pmpolicy log -r 3`

masterstatus	Reports the status of the production copy of the policy used by Privilege Manager for Unix to authorize commands.
	`masterstatus [-o <outfile>] [-c]`
	The production copy is stored in the following directory by default:
	`/etc/opt/quest/qpm4u/policy/`
	You can use the `-o` option to report the information to a file instead of to `stdout`.
	It reports the following information:
	• Path to the production copy
	• Date and time the production copy was checked out
	• Revision number of the production copy
	• Latest trunk revision number of the repository
	• Locally modified flag (indicates that someone manually edited the file)
	The information is displayed in the following format for CSV output:
	`<path>,<YYYY/MM/DD>,<HH:MM><policyrevision>,<trunkrevision>,0|1`

remove	Removes a file from the specified path in the policy repository. `remove -p path -d dir [-n [-l <commitmsg>]] [-c] [-u <user>]` Removes a file from the indicated working copy directory. Use the `-p` option to specify a path to the file (relative to the top-level directory in the policy). Use the `-d` option to specify the directory of the working copy. The `-n` option commits the changes to the repository. If you use the `-n` option, you can also use the `-l` option to provide a commit log message. If you use `-n` without `-l`, the command interactively prompts you for the commit log message.
revert	Reverts to the selected revision of the policy.

Command	Description
	`revert [-c] [-r <version>][-l <commitmsg>]` Checks out a copy of the selected revision, edits the files, and checks the copy back in as the latest revision.

status	Verifies the working copy of the policy in the directory indicated.
	
	`status -d <dir> [-c]`
	
	Verifies the working copy of the policy in the specified directory. You can use this to verify the status of a working copy that was previously checked out, before attempting to commit any edits. Each file in the selected directory is checked against the latest version in the repository. For example:
	
	`pmpolicy status -d /tmp`
	
	Exit status codes:
	
	• 0: The working copy is up to date and has not been modified; no action is required.
	• 1: The working copy is up to date and has been modified; you must check in to commit the edits made in the working copy.
	
	To commit the changes, run:
	
	`pmpolicy commit -d <dir>`
	
	• 2: The working copy is out of date and has not been modified; You must check out to get an up-to-date copy of the policy before editing.
	
	To check out the latest copy, run:
	
	`pmpolicy checkout -d <dir>`
	
	• 3: The working copy is out of date and has been modified, but the changes do not conflict with the latest version. Therefore, a default check in will fail. To commit the you must use the `-a` option.
	
	To commit the changes, run:
	
	`pmpolicy commit -d <dir> -a merge`
	
	• 4: The working copy is out of date and has been modified and the changes conflict with the latest version, therefore a default check in will fail.
	
	To commit the changes and overwrite any conflicts with the

Command	Description
	working copy's changes run: `pmpolicy commit -d <dir> -a force` • 5: An error occurred when attempting to verify the status.
sync	Checks out the latest version to the production copy of the policy used by Privilege Manager for Unix to authorize commands. `sync [-f][-c]` Synchronize the local production copy of the policy with the latest revision in the repository.
-v	Displays the Safeguard version.

Related Topics

pmcheck

Sudo command is rejected by Safeguard for Sudo

Sudo policy is not working properly

pmpolicyplugin

Syntax

```
pmpolicyplugin [-c] -g | -h | -l | -s | -v
```

Description

Use the `pmpolicyplugin` command to display the revision status of the cached security policy on this host or to request an update from the central repository.

Options

`pmpolicyplugin` has the following options.

Table 34: Options: pmpolicyplugin

Option	Description
-c	Displays output in CSV, rather than human-readable format.

Option	Description
-g	Exports the latest copy of the policy to the production copy (equivalent to `pmpolicy sync` on a server).
-h	Displays usage information.
-l	Reports whether a client is configured on this host.
-s	Shows details of the production policy on this host (equivalent to `pmpolicy masterstatus` on a server).
-v	Displays Safeguard version number.

See Sudo policy is not working properly on page 74 for an example of using the `pmpolicyplugin` command.

pmpoljoin_plugin

Syntax

```
pmpoljoin_plugin -j <primaryserver> [-u <localuser][-b][-p] -d [-f] [-b] -v | -h
| -[-z on|off[:<pid>]]
```

Description

Adjunct program to the `pmjoin_plugin` script. `pmpoljoin_plugin` is called by the `pmjoin_plugin` script when configuring a Sudo Plugin host to setup up the required read-only access to the policy repository, so that the client can operate in off-line mode.

Options

`pmpoljoin_plugin` has the following options.

Table 35: Options: pmpoljoin_plugin

Option	Description
-b	Runs the script in non-interactive mode. Default: Runs in interactive mode.
-d	Unconfigures the client.
-f	Does not prompt for confirmation when unconfiguing the client.
-h	Shows this usage

Option	Description
-j <primaryserver>	Joins this client to the selected primary server. Configures a client license on this host if it does not already have a server license; creates a pmclient user and configures read-only access to the repository for this user, using the pmpolicy account on the primary server.
-q	Reads pmpolicy user's password from stdin.
-u <localuser>	Specifies the `pmclient` user account that will manage the production copy. This user will be created if it does not exist. Default: pmclient
-v	Prints the product version.

pmpolsrvconfig

Syntax

```
pmpolsrvconfig -p <policygroupname> [-b][-i <path>][-o][-r <dir>]
                [-t sudo|pmpolicy] [-u <policyuser][-w <userpasswd>]
                [-g <policygroup>][-l <loggroup>] -s <host> [-b][-q] [-q]
                 -a <user> [-b][-q] [-q]
                 -d [-f]
                 -e <host> [-f]
                 -x [-f]
                 -v
                 -h
```

Description

The `pmpolsrvconfig` program is normally run by `pmsrvconfig` script, not by the user, to configure or un-configure a primary or secondary policy server. But, you can use it to grant a user access to a repository.

Options

`pmpolsrvconfig` has the following options.

Table 36: Options: pmpolsrvconfig

Option	Description
-a <user>	Provides the selected user with access to the existing repository. If the

Option	Description
	user does not exist, it is created. The host must first have been configured as a policy server. This user will be added to the `pmpolicy` group to grant it read/write access to the repository files, and to the `pmlog` group to grant it read access to the log files. On a secondary policy server, an `ssh` key will also be generated to provide access to the `pmpolicy` user account on the primary policy server. The "join" password is required to copy this ssh key to the primary policy server.
-b	Runs the script in batch mode (that is, no user interaction is possible). Default: Runs in interactive mode.
-d	Unconfigures the policy server, and deletes the repository if this is a primary server. If you do not specify the `-f` option, then it prompts you to confirm the action.
-e <host>	Removes the selected host from the server group.
-f	Forces the unconfigure action (that is, no user interaction required) Default: Prompt for confirmation for `-x` option.
-g <policygroup>	Specifies the policy group ownership for the repository. If this group does not exist, it is created. Default: pmpolicy
-h	Prints help.
-i <path>	Imports the selected policy into the repository. If this is a directory, the entire contents of the directory will be imported. Default: `/etc/sudoers`.
-l <loggroup>	Specifies the `pmlog` group ownership for the keystroke and audit logs Default: pmlog
-o	Overwrites the repository if it already exists. Default: Does not overwrite if the repository already exists.
-p <policygroup>	Configures a primary policy server for the selected group name.
-q	Reads the pmpolicy user's password from stdin.
-r <dir>	Creates the repository in the selected directory. Default: `/var/opt/quest/qpm4u/.qpm4u/.repository`
-s <host>	Configures a secondary policy server. You must supply the primary

Option	Description
	policy server host name. The secondary policy server retrieves the details of the policy group from the primary policy server. It creates the `policygroup` and `loggroup` groups to match those on the primary policy server and configures the `policyuser` user to grant it `ssh` access to the repository on the primary server. The "join" password is required to copy this `ssh` key to the primary policy server.
-t sudo|pmpolicy	Specifies the security policy type: sudo or pmpolicy. Default: sudo policy type
-u <policyuser>	Specifies the policy user account that manages the production copy. If this user does not exist, it is created and added to both the `policygroup` and `loggroup` groups. This user owns the repository on the primary policy server and provides remote access to the repository files to the secondary policy servers. Default: pmpolicy
-v	Prints the product version.
-w <userpasswd>	(Optional) Sets new user's password for `-a` option. Default: No password is configured.
-x	Unconfigures the policy server. If you do not specify the `-f` option, you are prompted to confirm the action. This does not remove the repository.

pmremlog

Syntax

```
pmremlog -v
pmremlog -p pmlog|pmreplay|pmlogtxtsearch [-o <outfile>]
pmremlog [-h <host>] [-b] [-c] -- <program args>
```

Description

The `pmremlog` command provides a wrapper for the `pmlog` and `pmreplay` utilities to access the event (audit) and keystroke (I/O) logs on any server in the policy group. Anyone in the `pmlog` group can run this utility on the primary policy server.

Note that `pmlogtxtsearch` is a command located in `/opt/quest/libexec`.

Options

`pmremlog` has the following options.

Table 37: Options: pmremlog

Option	Description
-b	Disables interactive input and uses batch mode.
-c	Displays output in CSV, rather than human-readable format.
-h <host>	Specifies a host in the policy server group to access.
-o <outfile>	Saves the `pmlog` output to a file.
-p	Specifies program to run: • pmlog • pmreplay • pmlogtxtsearch
-v	Displays the Safeguard version number.

Examples

To view the audit log on the primary policy server, enter:

```
pmremlog -p pmlog -- -f /var/opt/quest/qpm4u/pmevents.db
```

To view the audit events for user **fred** on secondary policy server **host1**, save the `pmlog` output to a file, and display the result of the `pmremlog` command in CSV format, enter:

```
pmremlog -p pmlog -c -o /tmp/events.txt -h host1 -- --user fred
```

To view the `stdout` from keystroke log **id_host1_x3jfuy**, on secondary policy server **host1**, enter:

```
pmremlog -p pmreplay -h host1 -- -o -f /var/opt/quest/qpm4u/iologs/id_
host1_x3jfuy
```

To retrieve the contents of keystroke log **id_host1_x3jfuy**, from secondary policy server **host1**, formatted for the `pmreplay` GUI, save the output to a temporary file, and display the result of the `pmremlog` command in CSV format, enter:

```
pmremlog -p pmreplay -h host1 -c -o /tmp/replay -- -zz -f
/var/opt/quest/qpm4u/iologs/id_host1_x3jfuy
```

pmreplay

Syntax

```
pmreplay -V
pmreplay -[t|s|i] -[Th] <filename>
pmreplay -[e][I][o] -[EhKTv] <filename>
```

Description

Use the `pmreplay` command to replay a log file to review what happened during a specified privileged session. The program can also display the log file in real time.

When using Safeguard for Sudo, enable keystroke logging by configuring the `log_input` and `log_output` variables. Please consult your sudoers manual for more information about configuring keystroke logging.

`pmreplay` can distinguish between old and new log files. If `pmreplay` detects that a log file has been changed, a message displays to tell you that the integrity of the file cannot be confirmed. This also occurs if you run `pmreplay` in real time and the Safeguard session that generated the events in the log file is active; that is, the client session has not completed or closed yet. In this case, the message does not necessarily indicate that the file has been tampered with.

The name of the I/O log is a unique filename constructed with the `mktemp` function using a combination of policy file variables, such as `username`, `command`, `date`, and `time`.

Safeguard sets the permissions on the I/O log file so that only `root` and users in the pmlog group can read it. That way, ordinary users cannot examine the contents of the log files. You must be logged in as `root` or be a member of the pmlog group to use `pmreplay` on these files. You may want to allow users to use Safeguard to run `pmreplay`.

By default `pmreplay` runs in interactive mode. Enter **?** to display a list of the interactive commands you can use to navigate through the file.

For example, replay a log file interactively by typing:

```
pmreplay /var/opt/quest/qpm4u/iolog/demo/dan/id_20130221_0855_gJfeP4
```

the results will show a header similar to this:

```
 Log File : /var/opt/quest/qpm4u/iolog/demo/dan/id_20130221_0855_gJfeP4 Date :
2013/02/21 Time : 08:55:17 Client : dan@sala.abc.local Agent :
root@sala.abc.local Command : id Type ’?’ or ’h’ for help
```

Type **?** or **h** at any time while running in interactive mode to display the list of commands that are available.

Options

`pmreplay` has the following options.

Table 38: Options: pmreplay

Option	Description
-e	Dumps the recorded standard error.
-E	Includes vi editing sessions when used with `-K`.
-h	When used with `-o` or `-I`, prints an optional header line. The header is always printed in interactive mode.
-i	Replays the recorded standard input.
-I	Dumps the recorded standard input, but converts carriage returns to new lines in order to improve readability.
-K	When used with -e, -I, and -o, removes all control characters and excludes `vi` editing sessions. Use with `-E` to include `vi` editing sessions.
-o	Dumps the recorded standard output.
-s	Automatically replays the file in slide show mode. Use **+** and **-** keys to vary the speed of play.
-t	Replays the file in tail mode, displaying new activity as it occurs.
-T	Displays command timestamps.
-v	Prints unprintable characters in octal form (\###)
-V	Displays the Safeguard version number.

Exit codes

`pmreplay` returns these codes:

- 1: File format error – Cannot parse the logfile.
- 2: File access error – Cannot open the logfile for reading
- 4: Usage error – Incorrect parameters were passed on the command line
- 8: Digest error – The contents of the file and the digest in the header do not match

Navigating the log file

Use the following commands to navigate the log file in interactive mode.

Table 39: Log file navigation shortcuts

Command	Description
g	Go to start of file.

Command	Description
G	Go to end of file.
p	Pause or resume replay in slide show mode.
q	Quit the replay.
r	Redraw the log file from start.
s	Skip to next time marker. Allows you to see what happened each second.
t	Display time of an action at any point in the log file.
u	Undo your last action.
v	Display all environment variables in use at the time the log file was created.
Space key	Go to next position (usually a single character); that is, step forward through the log file.
Enter key	Go to next line.
Backspace key	Back up to last position; that is, step backwards through the log file.
/<Regular Expression> **Enter**	Search for a regular expression while in interactive mode.
/**Enter**	Repeat last search.

Display the time of an action at any point in the log file with `t`, redraw the log file with `r`, and undo your last action with `u`.

You can also display all the environment variables which were in use at the time the log file was created using `v`. Use `q` or `Q` to quit `pmreplay`.

Type any key to continue replaying the I/O log.

pmresolvehost

Syntax

```
pmresolvehost -p|-v|[-h <hostname>] [-q][-s yes|no]
```

Description

The `pmresolvehost` command verifies the host name / IP resolution for the local host or for a selected host. If you do not supply arguments, `pmresolvehost` checks the local host

name/IP resolution.

Options

`pmresolvehost` has the following options.

Table 40: Options: pmresolvehost

Option	Description
-h <hostname>	Verifies the selected host name.
-p	Prints the fully qualified local host name.
-q	Runs in silent mode; displays no errors.
-s	Specifies whether to allow short names.
-v	Displays the Safeguard version.

pmserviced

Syntax

```
pmserviced [-d] [-n] [-s] [-v]
```

Description

The Safeguard service daemon, (`pmserviced`) is a persistent process that spawns the configured Safeguard services on demand. The `pmserviced` daemon is responsible for listening on the configured ports for incoming connections for the Safeguard daemons.It is capable of running the `pmmasterd` service.

Only one of `pmmasterd` and `pmclientd` may be enabled as they use the same TCP/IP port. See the individual topics in PM settings variables on page 112 for more information about these daemon settings.

Options

`pmserviced` has the following options.

Table 41: Options: pmserviced

Option	Description
-d	Logs debugging information such as connection received, signal receipt and service execution.

Option	Description
	By default, pmserviced only logs errors.
-n	Does not run in the background or create a pid file. By default, `pmserviced` forks and runs as a background daemon, storing its pid in `/var/opt/quest/qpm4u/pmserviced.pid`. When you specify the `-n` option, it stays in the foreground. If you also specify the `-d` option, error and debug messages are logged to the standard error in addition to the log file or syslog.
-s	Connects to the running `pmserviced` and displays the status of the services, then exits.
-v	Displays the version number of Safeguard and exits.

pmserviced Settings

`pmserviced` uses the following options in `/etc/opt/quest/qpm4u/pm.settings` to determine the daemons to run, the ports to use, and the command line options to use for each daemon.

Table 42: Options: pmserviced

Daemon Name	Flag to enable daemon	Listen on port	Command line options
pmmasterd	pmmasterdEnabled	masterport	pmmasterdOpts

Table 43: Settings: pmserviced

Setting	Description
pmservicedLog pathname | syslog	Fully qualified path to the `pmserviced` log file or syslog.
pmmasterdEnabled YES | NO	When set to YES, `pmserviced` runs `pmmasterd` on demand.
masterport number	The TCP/IP port `pmmasterd` uses to listen.
pmmasterdOpts options	Any command line options passed to `pmmasterd`.

Files

- settings file: `/etc/opt/quest/qpm4u/pm.settings`
- pid file: `/var/opt/quest/qpm4u/pmserviced.pid`

Related Topics

pmmasterd

pmsrvcheck

Syntax

```
pmsrvcheck --csv [ --verbose ] | --help | --pmpolicy | --primary | --secondary
```

Description

Use `pmsrvcheck` to verify that a policy server is setup properly. It produces output in either human-readable or CSV format similar to that produced by the preflight program.

The `pmsrvcheck` command checks:

- that the host is configured as a primary policy server and has a valid repository
- has a valid, up-to-date, checked-out copy of the repository
- has access to update the repository
- has a current valid Safeguard license
- `pmmasterd` is correctly configured
- `pmmasterd` can accept connections

`pmsrvcheck` produces output in either human-readable or CSV format similar to the pre-flight output.

Options

`pmsrvcheck` has the following options.

Table 44: Options: pmsrvcheck

Option	Description
--cvs	Displays csv, rather than human-readable output.
--help	Displays usage information.
--pmpolicy	Verifies that Safeguard policy is in use by the policy servers.
--primary	Verifies a primary policy server.
--secondary	Verifies a secondary policy server.
--verbose	Displays verbose output while checking the host.
--version	Displays the Safeguard version number and exits.

Files

- Settings file: `/etc/opt/quest/qpm4u/pm.settings`

Related Topics

pmmasterd

pmsrvconfig

Checking the policy server

Sudo command is rejected by Safeguard for Sudo

pmsrvconfig

Syntax

```
pmsrvconfig -h | --help [-abipqtv] [-d <variable>=<value>] [-f <path>]
            [-l <license_file>]
            [-m sudo | pmpolicy] [-n <group_name> | -s <hostname>]
             [-bpvx] -u [--accept] [--batch]
            [--define <variable>=<value>] [--import <path>] [--interactive]
            [--license <license_file>]
            [--name <group_name> | --secondary <hostname>]
            [--pipestdin] [--plugin] [--policymode sudo | pmpolicy]
            [--unix [<policy_server_host> ...]]  [--verbose] [--batch]
         [--plugin] [--unix] [-- verbose] --unconfig -N policy_name [--
policyname policy_name]
```

Description

Use the `pmsrvconfig` command to configure or reconfigure a policy server. You can run it in interactive or batch mode to configure a primary or secondary policy server.

Options

`pmsrvconfig` has the following options.

Table 45: Options: pmsrvconfig

Option	Description
-a | --accept	Accepts the End User License Agreement (EULA), `/opt/quest/qpm4u/qpm4u_eula.txt`.
-b | --batch	Runs in batch mode; does not use colors or require user input.
-d <variable>=<value> | --define <variable>=<value>	Specifies a variable for the `pm.settings` file and its associated value.

Option	Description
-h | --help	Displays usage information.
-i | --interactive	Runs in interactive mode; prompts for configuration parameters instead of using the default values.
-f <path> | --import <path>	Imports policy data from the specified path. • Privilege Manager for Unix: The path may be set to either a file or a directory when using the pmpolicy type. • Safeguard for Sudo: The path must be set to a file when using the sudo policy type.
-l | --license <license_file>	Specifies the full pathname of an .xml license file. You can specify this option multiple times with different license files.
-m sudo | pmpolicy | --policymode sudo | pmpolicy	Specifies the type of security policy: • sudo • pmpolicy Default: sudo
-n | --name <group_name>	Uses `group_name` as the policy server group name.
-p | --plugin	Configures the Sudo Plugin. This option is only available when using the sudo policy type (Safeguard for Sudo).
-q | --pipestdin	Pipes password to stdin if password is required.
-s | --secondary <hostname>	Configures host to be a secondary policy server where hostname is the primary policy server.
-u | --unconfig	Unconfigures a Privilege Manager for Unix server.
-v | --verbose	Displays verbose output while configuring the host.
-N policy_name | --policyname policy_name	When configuring the plugin, use `policy_name` as the name of the policy instead of the default. This option is used to specify the name of the policy that the server should use when making policy decisions.

Examples

The following example accepts the End User License Agreement (EULA) and imports the sudoers file from `/root/tmp/sudoers` as the initial policy:

```
# pmsrvconfig -a -f /root/tmp/sudoers
```

By using the -a option, you are accepting the terms and obligations of the EULA in full.

By default, the primary policy server you configure uses the host name as the policy server group name. To provide your own group name, use the -n command option, like this:

```
# pmsrvconfig -a -n <MyPolicyGroup>
```

where <MyPolicyGroup> is the name of your policy group.

Files

Directory where pmsrvconfig logs are stored: /opt/quest/qpm4u/install

Related Topics

pmjoin_plugin

pmmasterd

pmpolicy

pmsrvinfo

Syntax

```
pmsrvinfo [--csv] | -v
```

Description

Use the pmsrvinfo command to display information about the group in either human readable or CSV format. You can run this program on any server in the policy group.

Options

pmsrvinfo has the following options.

Table 46: Options: pmsrvinfo

Option	Description
-c	Displays information in .CSV format, instead of human readable output.
-l	By using this option, you can detect which client uses which sudo policy on the policy server. This option lists the following client information from the policy server: • Client's hostname

Option	Description
	• Sudoers file used by the client • Client's version This option can be used together with the "-c" option.
-v	Displays the Safeguard version number and exits.

Examples

```
# pmsrvinfo
```

```
Policy Server Configuration:
----------------------------
Safeguard version   : 6.0.0 (nnn)
Listening port for pmmasterd daemon    : 12345
Comms failover method                  : random
Comms timeout(in seconds)              : 10
Policy type in use                     : sudo
Group ownership of logs                : pmlog
Group ownership of policy repository   : pmpolicy
Policy server type                     : primary
Primary policy server for this group   : adminhost1
Group name for this group              : adminGroup1
Location of the repository             :
file:////var/opt/quest/qpm4u/.qpm4u/.repository/sudo_repos/trunk
Hosts in the group                     : adminhost1 adminhost2
```

Related Topics

Policy servers are failing

Sudo command is rejected by Safeguard for Sudo

pmsum

Syntax

```
pmsum /<full_path_name>
```

Description

Use `pmsum` to generate a checksum of the named file. The output it produces can be used in a policy with the `runcksum` variable. If the requested binary/command does not match the checksum, it rejects the command.

Options

`pmsum` has the following options.

Table 47: Options: pmsum

Option	Description
-v	Prints the version number of Safeguard and exits.

Examples

```
# pmsum /bin/ls
5591e026 /bin/ls
```

pmsysid

Syntax

```
pmsysid [-i] | -v
```

Description

The `pmsysid` command displays the Safeguard system ID.

Options

`pmsysid` has the following options.

Table 48: Options: pmsysid

Option	Description
-i	Shows the system host name and IP address.
-v	Displays the Safeguard version and exits.

Appendix C

Installation Packages

Safeguard is comprised of the following packages:

- **Privilege Manager for Unix product**

  Contains the Privilege Manager for Unix Policy Server and PM Agent components and uses the native packaging system for each platform (RPM, PKG, etc).

- **Safeguard for Sudo product**

  Contains the Safeguard Policy Server and Sudo Plugin components and uses the native packaging system for each platform (RPM, PKG, etc).

- **Preflight Binary**

  This is a stand-alone native binary for each platform (not zipped, tarred or packaged). This binary exists stand-alone on the ISO to make it available for use prior to installing software. It does not change any Safeguard configuration on the host.

For more information, see Download Safeguard for Sudo software packages on page 15..

Package locations

Safeguard is provided in native platform install packages, which include binary files, online man pages, installation files, and configuration file examples.

The install packages are located in the zip archive in two directories called:

- /server
- /agent
- /sudo_plugin

where <platform> is the name of the platform on which you are running Safeguard.

There are three different packages:

- `qpm-agent` package, which contains only the client (`pmrun`) and agent (`pmlocald`) components for Safeguard for Sudo.

- `qpm-server` package, which contains the server (`pmmasterd`), the client (`pmrun`) and agent (`pmlocald`), and the Sudo Plugin (`qpm4u_plugin.so`) components for Safeguard.
- `qpm-plugin` package, which contains the offline policy cache server (`pmmasterd`), the Sudo Plugin (`qpm4u_plugin.so`) components for Safeguard.

The Solaris server and agent packages have filenames that start with `QSFTpmsrv` and `QSFTpmagt`, respectively.

Once installed, the packaged files are placed in an installation directory under `/opt/quest` which contains subdirectories and files.

The platform directories contain the Safeguard installer packages for each platform supported by Safeguard.

Table 49: Privilege Manager kit directories

Platform	Architecture
aix71-rs6k	IBM®AIX 7.1, 7.2
freebsd-x86_64	FreeBSD on x86 64-bit architecture
hpux-hppa11	HP-UX 11.31 PA-RISC architecture
hpux11-ia64	HP-UX 11.31 Itanium architecture
linux-aarch64	Linux on ARM 64-bit architecture
linux-ia64	Linux on Itanium architecture
linux-intel	Linux x86
linux-ppc64	Linux on ppc little endian 64-bit architecture
linux-ppc64le	Linux on ppc little endian 64-bit architecture
linux-s390	Linux s390
linux-x86_64	Linux on x86 64-bit architecture
macos-x86_64	macOS on x86 64-bit architecture
Solaris-intel	Solaris Intel architecture
Solaris-sparc	SolarisSPARC® architecture

Installed files and directories

The following table lists files and directories installed on your system.

Table 50: Installed files and directories

Directories and files	Description	Created by
/opt/quest/qpm4u	Install directory containing readme, default trial license file, examples directory, templates, etc.	INSTALL
/etc/opt/quest/qpm4u/pm.settings	Configuration file for Safeguard component communications.	CONFIG
/etc/opt/quest/qpm4u/policy/pm.conf	Default production policy file when using the pmpolicy policy type.	CONFIG
/etc/opt/quest/qpm4u/policies	Default production policy framework directory when using the pmpolicy type.	CONFIG
/etc/opt/quest/qpm4u/policies/sudoers	Default production policy file for the sudo policy type.	CONFIG
/opt/quest/bin	Install directory containing the binaries for user programs, such as `pmrun`, `pmksh` and `pmvi`. These user programs only apply to Safeguard for Sudo.	CONFIG
/opt/quest/sbin	Install directory containing the binaries for admin programs, such as `pmlog` and `pmreplay`.	INSTALL
/opt/quest/lib	Install directory for shared libraries	INSTALL
/opt/quest/libexec	Install directory for dynamically loaded objects.	INSTALL
/opt/quest/man	This directory contains all the man pages for Safeguard daemons and programs.	INSTALL
/opt/quest/qpm4u/examples	This directory contains useful programs, scripts, or examples which show how to use Safeguard for Sudo. It also contains a sample configuration file which you	INSTALL

Directories and files	Description	Created by
	can use as a template for implementing your own policies. These scripts and examples only apply to Safeguard for Sudo.	
/opt/quest/qpm4u/license	This file contains the license information (policy server only). For information about updating license information, see pmlicense on page 143.	INSTALL
/opt/quest/qpm4u/qpm4u_eula.txt	This file contains the End User License Agreement for the Safeguard product.	INSTALL
/opt/quest/qpm4u/README. <architecture>	This file contains the latest information about your version of Safeguard.	INSTALL
/var/opt/quest/qpm4u/iolog	This directory contains the keystroke logs.	EVENTDATA
/var/opt/quest/qpm4u/pmevents.db	This file contains the event logs.	EVENTDATA

Appendix D

Unsupported Sudo Options

Sudo Plugin supports all sudo command options except those listed in the following tables:

- Unsupported command line sudo options
- Behavioral change
- Unsupported Sudoers policy options
- Unsupported Sudoers directives

Unsupported command line sudo options

Table 51: Unsupported command line sudo options

Sudo option	Description
-a <type>	Uses the specified authentication type.
-c <class>	Runs the specified command with resources limited by the specified login class.
-ll	Lists allowed commands in long format.
-r <role>	Causes security context to have specified role. SELinux RBAC is not supported.
-t <type>	Causes security context to have specified type. SELinux RBAC is not supported.

Behavioral change

Table 52: Behavioral change

Sudo option	Description
-k and -K	These flags only remove the user's credentials within the cache.
env_file	When in "offline policy evaluation" mode, this option only works if the file is present on the off-line host.
fqdn	Normally, when a policy has this flag enabled, sudo resolves host names on the policy server. However, when in off-line mode, sudo resolves host names from the policy cache server, which may produce different results.
group_ plugin	When in "off-line policy evaluation" mode, this option only works if the off-line host has `group_plugin` in the same path as the primary/secondary server.
lecture_ file	When in "off-line policy evaluation" mode, this option only works if the file is present on the off-line host.
logfile	When in "off-line policy evaluation" mode, this option only works if the file is present on the off-line host.

Unsupported Sudoers policy options

Table 53: Unsupported Sudoers policy options

Sudoers option	Explanation
compress_io	Compresses I/O logs using `zlib`.
fast_glob	`fast_glob` is always enabled; disabling `fast_glob` has no effect.
ignore_local_sudoers	Sudoers in LDAP is not supported.
iolog_dir ('%') escape sequences %{seq}	The %{seq} escape sequence is not supported.
iolog_flush	Safeguard keystroke logs are not buffered so this option is always on.
iolog_group	Safeguard keystroke logs are owned by the pmlog group.
iolog_mode	Safeguard keystroke logs are readable and writable by the root user and readable by the pmlog group.
iolog_user	Safeguard keystroke logs are owned by the root user.

Sudoers option	Explanation
limit-privs	Default set of Solaris limit privileges; not supported.
maxseq	Maximum I/O sequence number; not used by Safeguard.
pam_acct_mgmt	Safeguard always runs PAM account management modules.
pam_login_service	PAM service to use for login shells; Safeguard always uses "sudo".
pam_service	PAM service name to use; Safeguard always uses "sudo".
pam_session	Safeguard always creates a new PAM session.
pam_setcred	Attempts to establish PAM credentials for the target user; not used by Safeguard.
passprompt_override	Forces sudo to always use `passprompt`.
privs	Default set of permitted Solaris privileges, not supported.
pwfeedback	When set, sudo provides visual feedback when you press a key.
role	SELinux RBAC not supported.
stay_setuid	Forces sudo to act as a `setuid` wrapper.
timestamp_type	Safeguard uses its own time stamp format.
timestampdir	The directory in which sudo stores its timestamp files.
timestampowner	The owner of the `timestamp` directory and the timestamps stored therein.
type	SELinux RBAC not supported.
use_pty	Not relevant; `pty` is always used.

Unsupported Sudoers directives

Table 54: Unsupported Sudoers directives

Sudoers directive	Description / Explanation
#include & #includedir	Safeguard for Sudo does not support these options. Because these options use absolute paths they can point outside the policy repository making it impossible to sync the policy files that are included among the policy servers. You can use #include and #includedir to include files and directories, so

Sudoers directive	Description / Explanation
	long as you keep them in a separate directory from the working copy, but you need to know that the included files/directories will not be under revision control.

Appendix E

Safeguard for Sudo Policy Evaluation

Safeguard for Sudo enhances traditional sudo by providing centralized services for policy evaluation, as well as event and keystroke logging. Safeguard for Sudo provides continuity of service in the event of a network or server outage by means of off-line policy evaluation.

Sudo off-line policy evaluation is available when using the Sudo Plugin joined to a policy server that evaluates a sudoers policy.

How it works

The Sudo Plugin package provides a cache service by installing a client version of the policy server daemon (`pmmasterd`) on the Plugin host. When you configure and join the host to a policy server, it sets up the policy management subsystem, and checks out the cache's copy of the security policy from the central repository on the primary policy server.

When you run a sudo command, it sends the initial Sudo Plugin request to the cache service running on the Plugin host. In most cases, the cache service forwards the request to a central policy server on the network. However, if the cache service does not receive a timely response from a central policy server, it services the request locally, performing an off-line evaluation of the cached policy and storing the event and keystroke logs in a temporary holding location on the Plugin host (under `/var/opt/quest/qpm4u/offline/`).

You can configure the time period before an off-line policy evaluation occurs by adding the `offlineTimeout` setting in the `/etc/opt/quest/qpm4u/pm.settings` file. `offlineTimeout` defaults to 1500 milliseconds (1.5 seconds). To modify that setting, specify the timeout period in milliseconds as an integer value. For example, to set a timeout of 5 seconds (5000 milliseconds), enter the following into the `pm.settings` file:

```
offlineTimeout 5000
```

Setting `offlineTimeout` to 0 in the `pm.settings` file, forces the cache service to always perform off-line (local-only) policy evaluation for sudo requests.

Once an off-line policy evaluation has occurred, the `pmloadcheck` daemon periodically checks to see if any policy server has come back online. Upon returning to an online state, the `pmloadcheck` daemon initiates a log file transfer to copy the logs to a temporary quarantine area on the policy server (`/var/opt/quest/qpm4u/quarantine`). The policy server

validates the integrity of the log files in the quarantine and processes them, appending events to the central log store.

Determining off-line events

When off-line policy evaluation occurs, the cache service's `pmmasterd` process writes a message to the `pmmasterd.log` file located in either `/var/log` or `/var/adm`, depending on your operating system, and is configurable in the `/etc/opt/quest/qpm4u/pm.settings` file.

Once processed into a policy server's central event store, you can identify off-line events by examining the `offlinesession` event variable (`pmlog -c "offlinesession==1"`) or the `masterhost` variable which is set to the Plugin host's hostname.

Off-line policy cache updates

At regular intervals and whenever a request is sent to a central policy server for online evaluation, the `pmloadcheck` daemon checks the revision number of the cached policy. You can also use the `pmpolicyplugin` utility to display the revision status of the cached policy or to request an update. See pmpolicyplugin on page 172 for more information about this utility.

About us

One Identity solutions eliminate the complexities and time-consuming processes often required to govern identities, manage privileged accounts and control access. Our solutions enhance business agility while addressing your IAM challenges with on-premises, cloud and hybrid environments.

Contacting us

For sales and other inquiries, such as licensing, support, and renewals, visit https://www.oneidentity.com/company/contact-us.aspx.

Technical support resources

Technical support is available to One Identity customers with a valid maintenance contract and customers who have trial versions. You can access the Support Portal at https://support.oneidentity.com/.

The Support Portal provides self-help tools you can use to solve problems quickly and independently, 24 hours a day, 365 days a year. The Support Portal enables you to:

- Submit and manage a Service Request
- View Knowledge Base articles
- Sign up for product notifications
- Download software and technical documentation
- View how-to videos at www.YouTube.com/OneIdentity
- Engage in community discussions
- Chat with support engineers online
- View services to assist you with your product

Index

J

K

L

M

N

O

P

R

S

T

U

V