SignServer
PKI by PrimeKey

SignServer TimeMonitor Manual
SignServer v 4.1.0

SignServer
PKI by PrimeKey

Table of Contents

SignServer v 4.1.0

1 Time Monitor Introduction

SignServer implements the Time-Stamp Protocol as specified in RFC #3161 and can be used as the core part of a Time Stamping Authority (TSA).

Using the correct time from a reliable time source is critical for the operation of a TSA. This document addresses this by describing the use of time synchronization using the Network Time Protocol (NTP), a special TimeSource in SignServer as well as an application called SignServer TimeMonitor.

1.1 Time requirements

Two specifications that put requirements on the TSA time management are:

1. RFC #3161 which specifies that the TSA is REQUIRED
    a. to use a trustworthy source of time.
    b. to include a trustworthy time value for each time-stamp token
2. ETSI TS 102 023 - V1.2.2 (2008-10) - Electronic Signatures and Infrastructures (ESI); Policy Requirements for time-stamping authorities which specifies that:
    a. "The TSA shall ensure that its clock is synchronized with UTC within the declared accuracy."
    b. "The calibration of the TSU clocks shall be maintained such that the clocks shall not be expected to drift outside the declared accuracy."
    c. "The TSA shall ensure that, if the time that would be indicated in a time-stamp token drifts or jumps out of synchronization with UTC, this will be detected..."
    d. "Records concerning all events relating to synchronization of a TSU's clock to UTC shall be logged. This shall include information concerning normal re-calibration or synchronization of clocks use in time-stamping. "
    e. "Records concerning all events relating to detection of loss of synchronization shall be logged. "

1.2 Fulfilling the requirements

The local clock of the server is synchronized with a reliable time source (i.e. time server with reliable clock synchronized with a national time source or GPS) using an NTP service provided by the operating system.

Calibration is performed using NTP and is therefor not expected to drift outside declared accuracy compared to the reliable time source.

The accuracy of the calibration is periodically monitored. A time-stamp token will not be issued unless the monitoring reported the the time to be in sync and the report was made within a configured interval (for instance 1 second).

Regarding logging:

- The time server should log its own events such as loss of connection with GPS etc.
- Operating system NTP service should log its events.
- Manual re-calibration should be logged (in operating system log or manually if no such logging exists).
- Monitoring tool should log when time is detected to be out of synchronization.
- SignServer/TimeStampSigner logs whether the time was considered in sync or not when processing a request.

2 Time sources in SignServer

A time-stamp signer in SignServer acquires the current time through its configured *TimeSource*. A *TimeSource* in SignServer is an implementation responsible for returning the time (if available). The time-stamp signer calls the *TimeSource* for every request to get the current time.

If the current time could not be acquired from the *TimeSource* the time-stamp signer will not issue the time-stamp token and instead respond to the signing request with the "Time source is not available" failure message.

2.1 Local computer time source

The default implementation is called the *LocalComputerTimeSource* which gets the time from the operating system. Using this *TimeSource* the time is always available, however it relies on the time as configured on the local server which might not be synchronized with a reliable time source. This *TimeSource* will not detect if the time jumps or drifts out of synchronization.

2.2 Status reading local computer time source

Another *TimeSource* implementation is the *StatusReadingLocalComputerTimeSource* which also gets the time from the operating system but only if the time is considered to be in sync with a reliable time source.

In SignServer there is a component called the *StatusRepository* which contains a set of pre-defined named properties. Each property can have a value and an optional expiration time. The value of status properties (and their expiration) can be updated by external applications/script using the command line interface or by calling a special worker. Signers can query the *StatusRepository* for the value of a property that has not yet expired.

The TimeSource will query a property called TIMESOURCE0_INSYNC and if it has the value 'true' and it is not expired, the time-source will consider the time to be in sync with a reliable time source.

It is the responsibility of some external application/service to monitor the status of the local time and update the status property accordingly. By setting the status to 'in sync' the external application asserts that it has checked that the local time was synchronized with some configured accuracy with an reliable time source at the time the property was updated. By also setting an expiration time on the value the application can indicate a maximum time the assertion is valid for and the application is expected to update the value before that. This assures that if the monitoring application fails for any reasons the value will expire and no more time-stamp tokens be issued until it is back.

It is assumed that the monitoring application performs its checks of the local time and updates the status properties often as this has a direct impact on the number of time-stamp tokens that could be incorrectly issued after a loss of time synchronization has happened.

3 SignServer TimeMonitor

The SignServer *TimeMonitor* is an implementation of the type of application that can be used together with the *StatusReadingLocalComputerTimeSource* for monitoring of the local time and informing SignServer about its state.

The application is started separately from SignServer and runs in its own process. The important part is the main loop which repeatably performs the following steps:

1. Calculate the time difference between the local time and the time of the time server by invoking the 'ntpdate' command.
2. The result is compared with the configured allowed time difference and if the time is within the interval the time is considered in sync.
3. The status is then published to SignServer using HTTP. The status is stored with an expiration time and it is therefor important that the application gets time to perform a new round and publish updated results before the expiration.

Changes to the TimeMonitor states are logged using Log4j.

The current status of the TimeMonitor can be queried using HTTP from a Health check page.

3.1 System overview
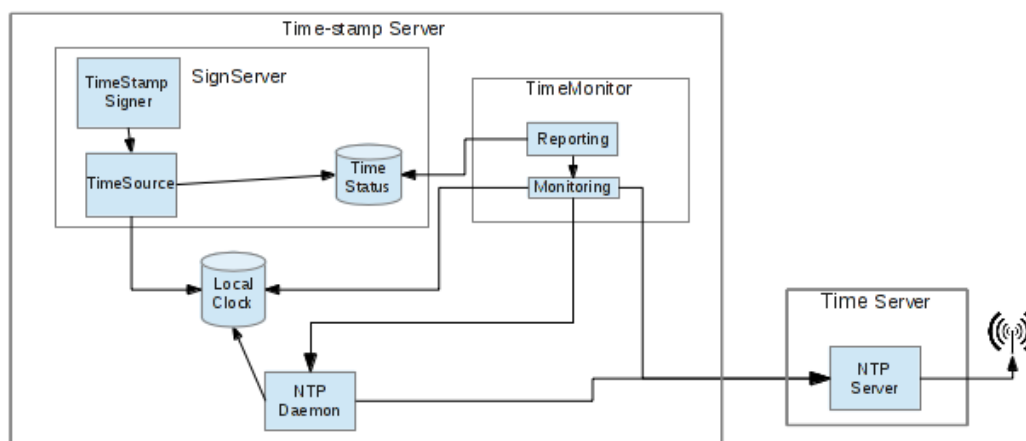
Picture: Interactions between SignServer, TimeMonitor and Time Server. Logging and general monitoring is left out.

3.2 Assumptions

- The local time of the server is synchronized with an external time server using for instance the NTP daemon.
- The external time server is synchronized with a reliable national time source or GPS.
- The external time server has an NTP service that can be queried often from the TimeMonitor application on each server.

- The time-stamp signer(s) uses the StatusReadingLocalComputerTimeSource.

Note: The TimeMonitor application can be configured to query the time server more often than the NTP standards allow (i.e. more often than every 15 seconds). It is therefor important not to run the TimeMonitor application against an online NTP service. The TimeMonitor application must only be run towards a dedicated NTP server under your own control.

3.3 Installation

The application is built with SignServer by setting the property "timemonitor.enabled=true" in conf /signserver_deploy.properties and issue the command:
$ bin/ant build

3.4 Configuration

The configuration can be done in conf/timemonitor.properties.

Note: The conf/ folder is included on the classpath when running the application using the wrapper script *bin/timemonitor-in-background.sh*. If the application is run some other way, edit src/timemonitor. properties or make sure conf/ is included in the classpath.

The 'application properties' listed in the next section are required to be specified in the properties file. The other set of properties called 'runtime properties' can either all be specified in the property file or be queried from a TimeMonitorManager worker in SignServer.

3.4.1 Application properties

Configuration property	*Description*
timeserver. ntpdatecommand	Path to the nptdate command executable. Sample value: "/usr/sbin/ntpdate"
timeserver. ntpqcommand	Path to the ntpq command executable Sample value "/usr/bin/ntpq"
timemonitor.stateweb. enabled	Set to "true" to enable (or "false" to disable) the state web server (Health check). Sample value: "true"
timemonitor.stateweb. bindaddress	IP address the server will bind to. Use "0.0.0.0" to bind to all interfaces and give external access to the server. Sample value: "127.0.0.1"

timemonitor.stateweb. port	TCP port to offer state information (HTTP) on. Sample value: "8980"
timemonitor.stateweb. threads	Number of threads in the thread pool handling incoming connections. Sample value: "5"
timemonitor.stateweb. backlog	Maximum number of queued incoming connections to allow. Incoming queued connections exceeding this limit may be rejected. If 0 is specified a system default value is used. Sample value: "0"
signserver.process.url	URL to the SignServer process that will handle the status update. Sample value: "http://localhost:8080/signserver/process"
signserver. statuspropertiesworker. name	Name of the *StatusPropertiesWorker* that will handle the status update. Note: SignServer will have to be configured with a either a *StatusPropertiesWorker or a TimeMonitorManager* with this name. It needs to use an Authorizer which gives the TimeMonitor access. For instance AUTHTYPE=NOAUTH can be used but that would also give everybody permission to update the status. Instead as the TimeMonitor is running on the same host as SignServer it is recommended to use an RemoteAddressAuthorizer and only allow requests from localhost. Sample value: "StatusPropertiesWorker"
signserver. statusproperty.name	Name of the status property to update for time synchronization status. Sample value: "TIMESOURCE0_INSYNC"
signserver. leapstatusproperty. name	Name of the status property to update for leap second status. Sample value: "LEAPSECOND"
signserver. managedconfig	When set to "true" the 'runtime properties' in the next section are not allowed in the configuration file but are instead queried from a TimeMonitorManager in SignServer. In this mode the TimeMonitor will start up disabled , meaning that the timeserver and NTP daemon will not be queried until the TimeMonitor gets an updated configuration from SignServer. In this mode the TimeMonitor will query SignServer every 15 seconds. Sample value: "true"

3.4.2 Runtime properties

The following additional properties can be specified in the configuration file if signserver. managedconfig=false or in the TimeMonitorManager if signserver.managedconfig=true.

Configuration property	Description
timeserver.host	Hostname or IP address of the time server that should be queried. Can also be a comma-separated list to use several servers as fallbacks. The ntpdate command will be called with all hosts as arguments and the first one returning a valid response will be used. Sample value: "192.168.20.10, 192.168.20.11"
timeserver. sendsamples	Number of samples (NTP packets) to send to the time server. This is the "-p" option of ntpdate and can be from 1 to 8 inclusive. Specifying a larger number of samples gives more accurate estimates but takes longer time to execute. Sample value: "2"
timeserver.timeout	Maximum wait time for response from the time server. This is the "-t" option of ntpdate. The unit is seconds but fractions rounded to a multiple of 0.2 are supported. Sample value: "0.2"
timemonitor. maxAcceptedOffset	Maximum difference (in milliseconds) for the local time as compared to the time server for the time status to still be INSYNC. Sample value: "997"
timemonitor. warnOffset	Difference (in milliseconds) for the local time as compared to the time server when the state changes to SOON_OUT_OF_SYNC. Sample value: "500"
timemonitor. statusExpireTime	Expire time (in milliseconds) to set when publishing the status to SignServer. Note: Make sure the TimeMonitor has enough time to run one round and publish a new value before the expiration otherwise SignServer will not be able to issue time-stamp tokens for a period of time. Sample value: "900"

timemonitor. leapStatusExpireTime	Expire time (in milliseconds) to set when publishing the leap second status to SignServer. Sample value: "60000"
timemonitor. minRunTime	Minimum time for one round by the TimeMonitor. If checking the time and publishing the status is performed in shorter time than this value (in milliseconds), TimeMonitor will sleep for the remaining time. Sample value: "500"
timemonitor. warnRunTime	If performing one round of checking the time and publishing the status takes longer time then this (in milliseconds) change the report state to REPORTED_BUT_EXPIRE_TIME_SHORT. This happening is an indication that the time settings needs to be adjusted. Sample value: "700"
timemonitor.disabled	When set to "true" the TimeMonitor is explicitly set in disabled mode, meaning that the timeserver and NTP daemon will not be queried until the TimeMonitor gets an updated configuration from SignServer. In this mode the TimeMonitor will query SignServer every 15 seconds.

3.5 Usage

The application can be started to run in background by issuing:

```
$ bin/timemonitor-in-background.sh
```

Watch **signserver-timemonitor.log** for output.

3.6 Logging

Logging is configured in **conf/log4j.properties** (given that the wrapper scripts in **bin/** is used to run the application).

By default events with the **INFO** level is logged to a local file called signserver-timemonitor.log.

It is recommended to use syslog or similar mechanisms to send the logs to a remote server where they can be inspected when the time is detected to be out of sync.

The TimeMonitor maintains three different types of states: **Time state**, **Report state**, and **Leap state**.

Time State for time synchronization describes the status of the time source and can be:

- **INSYNC**: The time is in sync as it was detected to be within the configured range.
- **SOON_OUT_OF_SYNC**: The time is in sync but was detected to be withing the configured range to give a warning.
- **OUT_OF_SYNC**: The time was detected to be out of sync.

- **UNKNOWN**: The status of the time is unknown as the time server has not yet been contacted, it could not be contacted or that some other error occurred preventing the TimeMonitor from getting the status.

Report State describes the status of the publishing of the results to SignServer and can be:

- **REPORTED**: The results were successfully published to SignServer.
- **REPORTED_BUT_EXPIRE_TIME_SHORT**: The results were successfully published to SignServer but the time it took to perform the measurements and publish it was longer than the time configured as timemonitor.warnRunTime.The log gives more information about the actual run time and how much time was spent during query and publishing when the state changes to this state.
- **FAILED_TO_REPORT**: The results could not be published to SignServer.An error message could be available in the log when the state changes to this state.

Leap State is reported to SignServer with the following values:

- **NONE**: No leap second is scheduled at the next possible leap second occurrence.

- **POSITIVE**: A positive (inserted) leap second is scheduled at the next possible leap second occurrence.

- **NEGATIVE**: A negative (removed) leap second is scheduled at the next possible leap second occurrence.

- **UNKNOWN**: Leap second state was unknown when querying the NTP daemon.

See also the section on StatusReportingLocalComputerTimeSource in SignServer about setting up leap second handling.

3.6.1 Examples

Each time any of the states changes the new Time, Report, and Leap states are logged at INFO level:

```
14:56:01,983  INFO TimeMonitorRunnable:60 - Started
14:56:02,491 ERROR TimeMonitorRunnable:214 - Command failed (1): 31 Oct 14:56:02
 ntpdate[26321]: no server suitable for synchronization found
14:56:02,511 ERROR TimeMonitorRunnable:268 - Failed to update status property: Connection
 refused
14:56:02,515  INFO TimeMonitorRunnable:91 - State changed to: UNKNOWN,FAILED_TO_REPORT,NONE
14:58:11,152  INFO TimeMonitorRunnable:91 - State changed to: UNKNOWN,REPORTED,NONE
15:01:23,310  INFO TimeMonitorRunnable:91 - State changed to: INSYNC,REPORTED,NONE
```

When the state changes from INSYNC or SOON_OUT_OF_SYNC to either OUT_OF_SYNC or UNKNOWN an additional log entry also outputs the last time that was determined to be in sync:

```
15:32:05,990 INFO  [TimeMonitorRunnable] State changed to: INSYNC,REPORTED,NONE
15:32:06,492 INFO  [TimeMonitorRunnable] State changed to: OUT_OF_SYNC,REPORTED,NONE
15:32:06,492 INFO  [TimeMonitorRunnable] Last trusted time was: 2012-11-27 15:32:05,990
```

3.7 Monitoring

The current state of the TimeMonitor application can also be monitored from an external service if the state-showing web (Health check) server is enabled. The state web page returns the current state as a comma separated list of values.

1. Update time: The current time when the expiration time to set was calculated
2. Time state: See states under Logging section
3. Report state: See states under Logging section
4. Leap state: See states under Logging section
5. Config version: Identifier for the last configuration received
6. Time offset: The measured difference in time with the time server
7. Time server query time: The time it took to query the time server
8. NTP daemon query time: The time it took to query the leap state
9. Report time: The time it took to report the status to SignServer.

Example getting the current state:

```
$ curl http://tsaserver:8980/state
 1409050686281,INSYNC,REPORTED,NONE,1ccdf46b,0,508,8,6
```

3.8 Rate-limiting

If an NTP server responds with a rate-limiting "kiss of death" response, the time monitor will log an error and stop querying and the state will be set to UNKNOWN. The time monitor will resume querying on the next configuration update (i.e. setting another NTP server host).

4 Testing and Troubleshooting

4.1 Guide to setting up a Linux server as lab time server

This section describes the steps to install and configure NTPd on a Linux server that can then be used to provide a "simulated" time to other test servers.

1. Make sure NTPd is installed

```
$ sudo yum install ntp
```

2. Configure the server to use its own time by making sure the configuration only has the server "server 127.127.1.1" configured in /etc/ntp.conf.

```
$ sudo vi /etc/ntp.conf
```

Information on the "undisciplined local clock" driver:
http://www.clock.org/ntp/driver1.html

3. As this time server has no connection with upstream timeservers or a GPS source we need to manually configure the information about leap seconds. This can be done by downloading the NIST leap seconds file from ftp://time.nist.gov/pub/leap-seconds.list and place it on the server:

```
$ sudo cp leap-seconds.list /var/lib/ntp/leap-seconds.list
```

4. Configure NTPd to use the leap seconds file by editing /etc/ntp.conf and specifying the path to this file:

```
$ sudo vi /etc/ntp.conf

 leapfile "/etc/leap-seconds.list"
```

Information on NTPd and NIST Leap Second File:
http://support.ntp.org/bin/view/Support/ConfiguringNTP#Section_6.14.

5. Set the time you want to simulate and restart NTPd:

```
$ sudo date --utc --set "2015-06-30 23:30"
$ sudo service ntpd restart
```

6. Check that the local clock was configured correctly and that the leapfile was picked up by querying all configured peers (time sources) and then checking the leap second variables and finally printing the current time just to be sure it is correct:

```
$ ntpq -p && ntpq -c "rv 0 leap,leapsec,tai" && date --utc
 remote refid st t when poll reach delay offset jitter
 ==============================================================================
 *LOCAL(1) .LOCL. 5 l 3 64 1 0.000 0.000 0.001
 leap=01, tai=35, leapsec=201507010000
 Tue Jun 30 23:30:11 UTC 2015
```

Comments

- Notice, the star before **LOCAL** meaning it it synchronized with this source.
- The refid **.LOCL.** meaning the local clock is used
- the **tai** and **leapsec** values are available meaning that the leap seconds file was read.
- the leap value ("leap indicator") here stating 01 meaning a positive leap second will be introduced at the end of the month.

In case you did not get the "tai" and "leapsec" variables printed the issue could be that the leap seconds file was not found or the system did not allow NTPd to read it. Check if you get any error in syslog.

In Debian/Ubuntu you might get an error from Apparmor like this:

```
kernel: [ 6327.597127] type=1400 audit(1432630950.384:12): apparmor="DENIED" operation="open"
 parent=1 profile="/usr/sbin/ntpd" name="/etc/leap-seconds.list" pid=2224 comm="ntpd"
 requested_mask="r" denied_mask="r" fsuid=0 ouid=0
```

In that case you will have to modify Apparmor to allow NTPd to read this location or place the file in a place which is already allowed. See: http://askubuntu.com/questions/571839/leapseconds-file-permission-denied.

7. Test servers could now be pointed to use this time server as its server by specifing its address in their /etc/ntp.conf. Remember to also point the SignServer TimeMonitor to this test time server. After changing the NTP configuration of the client its NTPd needs to be restarted and then you will need to wait until the time synchronizes. See the output from "ntpq -p" until the IP address of the time server gets a star in front of it, usually at reach 17 or at latest 377.