TimeKeeper version 8.0.31

Get started

Everything you need to get going with TimeKeeper.

TimeKeeper capabilities

TimeKeeper is a timing client, boundary clock, server, compliance tool, and more.

Features

An overview of the TimeKeeper feature set.

Grandmasters, Hardware

Grandmasters running TimeKeeper provide high quality NTP, PTP, and more.

Compliance

TimeKeeper Compliance analyzes and consolidates data for regulatory reporting.

Using TimeKeeper

Learn what tools TimeKeeper provides, specs about how it integrates, and details on all of its options.

Installation requirements

Packaging and platform details for TimeKeeper deployments.

Release notes

Details on updates and improvements to TimeKeeper.

TimeKeeper technology

Before diving into the specifics of different TimeKeeper features, let's take a quick look at what it is and what it does.

TimeKeeper is a flexible tool to synchronize local system time with many different possible time sources and optionally distribute that very accurate time many ways to many clients. It also monitors the time accuracy to alert when it's bad, take corrective action, record time accuracy and report on it. It is able to:

- Act as an NTP server and client
- Be a PTP Grandmaster, boundary clock, client
- Send SNMP traps, handle SNMP queries, emails, syslog notification of configured events
- Interop, out of the box, with other hardware and software solutions
- Support PCI/PCIe GPS/PPS/IRIG hardware
- High availability and transparent failover
- Provide simple centralized management of timing networks
- Web-based management, configuration, and visualization
- Generate compliance and regulatory reporting and data governance reports

The TimeKeeper solution

TimeKeeper tracks (and serves) all of the common timing protocols, but does so in a new way - accurately, regardless of protocol, all in the same system, all at the same time. TimeKeeper can track a GPS input, dozens of NTP servers, many PTP Grandmasters across different domains and interfaces, while at the same time serving both NTP and PTP to thousands of clients. It simultaneously maps the timing network and collects client timing details, building a consolidated audit trail of time across the network.

TimeKeeper automatically leverages hardware capabilities it can find. While tracking time sources, it will make use of hardware timestamping cards (Solarflare, Mellanox, Intel, Broadcom, etc) and oscillators if they're available. If hardware provides oscillators that can be steered, TimeKeeper will automatically and accurately steer them. TimeKeeper is designed to easily manage timing resources for you so that you don't have to invest countless hours building and maintaining a complex network of disparate components.

TimeKeeper Compliance provides a clear and accurate historical view into your timing network and the performance of your systems. It generates reports suitable for use in regulatory reporting and compliance.

We'll cover these features in detail as we go, but as an introduction, TimeKeeper uses the concept of 'time sources' that are specified in the TimeKeeper configuration. These sources may be some NTP servers, a set of PTP feeds from a group of Grandmasters, and a local PCIe device with a GPS antenna for example.

How TimeKeeper works

TimeKeeper is designed around the idea of 'sources'. It can be configured with up to 50 different time sources which can be NTP servers, PTP servers, GPS clocks or any variety of other ways to get time. Each time source is listed in the "timekeeper.conf" file like so with lower valued numbers being higher priority:

```
SOURCE0() { NTPSERVER=hostname1; }
SOURCE1() { NTPSERVER=hostname2; }
SOURCE2() { NTPSERVER=hostname3; }
```

TimeKeeper will keep track of all of these time sources at the same time but will only steer the system clock to one of these time sources which is considered the primary time source. This primary time source is picked with a priority system. In the example above "SOURCE0" would be used. When a time source stops providing time for 3 minutes it will be declared unavailable and TimeKeeper will fall back to the next time source in priority order. Once a time source begins responding again it will become the primary time source if it is highest priority. TimeKeeper is also able to cross-check time sources and declare some invalid for other reasons than simply not responding if configured to do so which is covered in the "Sourcecheck" section.

Installing TimeKeeper

Installation and setup of TimeKeeper is very simple:

- Run the installer with your license file in place
- Edit the configuration to set the time sources and options
- Start TimeKeeper
- Verify that TimeKeeper is working and running properly

(See the "Updating" section for instructions on how to update TimeKeeper and its license.)

On both Windows and Linux, the TimeKeeper web interface allows you to manage configuration, verify time sources, and more.

Install process

If you don't have TimeKeeper yet, contact FSMLabs or your FSMLabs' resellers for an evaluation. You'll be provided with the installer and a license file.

Once you've got your copy of the software and a license file (timekeeper.lic), put them both in the same directory and run the installer:

```
$ # Ensure the binary is executable
$ chmod +x timekeeper_installer*
$ ./timekeeper_installer*
```

> On Linux, the binary installer requires superuser privileges. It must be run as root.

> When downloading installers on some version of IE, it may add ".man" to the end of the file. Please remove the ".man" portion after downloading.

On Windows the installer can be run from File Explorer, and will ask for elevated privileges if needed.

Configure TimeKeeper

TimeKeeper's configuration is stored in */etc/timekeeper.conf* on Linux, and *%ProgramFiles%\timekeeper\timekeeper.conf* on Windows, which is normally *C:\Program Files\timekeeper\timekeeper.conf*.

There are a lot of reference options listed in the existing timekeeper.conf file to show different scenarios. Lines that start with a *#* are comments, and most of the file will be made up of these showing different configurations. Leave these sections as comments if you're not using them.

The first step is to set up at least one time source, which we'll call "source 0", and enable the web management interface, set up in the timekeeper.conf file like this:

```
SOURCE0() { NTPSERVER='ntp_server'; }
ENABLE_WEB_MANAGEMENT='1';
WEB_MANAGEMENT_PORT='8080';
```

This will make TimeKeeper get time via NTP from ntp_server as its primary source of time. You can specify many different time sources with different protocols as *SOURCE1*, *SOURCE2*, up to 50 sources, where the lowest source number is the highest priority source. We'll cover all these options in more detail later on.

It will also enable the web management interface, running on port 8080.

Start the service

Once TimeKeeper's installed and configured, the service is ready to be started. You can do that on Linux with:

```
$ systemctl start timekeeper
```

> Non systemd-based systems can be started similarly with *service timekeeper restart*.

On Windows, either start the service via the Windows Management Console, or from a command prompt running as Administrator with:

```
C:\> sc start timekeeper
```

Verify TimeKeeper is working

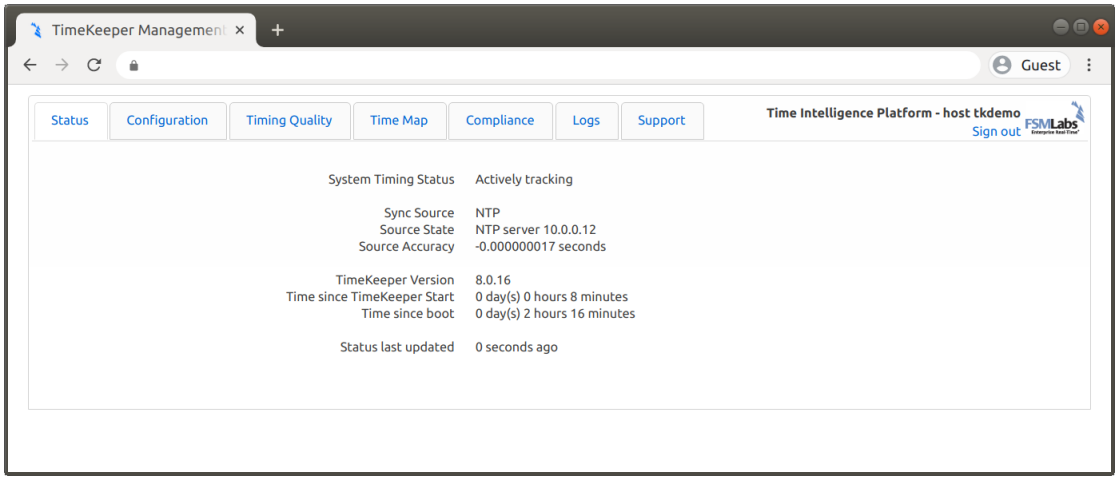
Now you can run *tkstatus* to get a quick overview of the system state, or connect to the web management interface (http://localhost:8080), and login with username "admin", default password "timekeeper". Note: In earlier versions of TimeKeeper the default password was "fsmlabs".

The tkstatus output may look similar to this:

```
TimeKeeper Status
================================================================================
TimeKeeper version 8.0.16
License is permanent.
Primary source is: 0
Primary source offset: 0.000000011 seconds
Primary source type: NTP(10.0.0.12:GPS)
Last time update: 1 seconds ago

Src|offset      |source type                |update age(s)|timestamp type
================================================================================
 0 +0.000000011             NTP(10.0.0.12:GPS)          1  TIMESTAMPING_HW
```

Similarly, the web status page will show:



You can also see the individual NTP time updates on the web interface under the "Timing Quality" tab, or in */var/log/timekeeper_0.data* on Linux, and *var\log\timekeeper_0.data* under the TimeKeeper installation on Windows. We'll cover those files and views in much more detail later, but at this point, you've got TimeKeeper running, and we can start looking at using the more advanced features.

Quick start TimeKeeper evaluation

The easiest way to test/evaluate TimeKeeper is to setup one computer as a server and another as a client, even if you have no existing time sources. The remainder of this section assumes you are starting with two Linux machines, on the same network and you have followed the "Installation" section.

Configuration

On the server, use the following *timekeeper.conf*:

```
SERVENTP=1
SOURCE0() { PPSDEV=self; }
SERVEPTP0() { PTPSERVERVERSION=2; }
ENABLE_WEB_MANAGEMENT=1
WEB_MANAGEMENT_PORT=8080
```

This will configure TimeKeeper to use the local system clock as a source and provide that time to clients on the network.

Then, on the client, edit *timekeeper.conf*:

```
SOURCE0() { NTPSERVER=IP_address_of_server; }
SOURCE1() { PTPCLIENTVERSION=2; }
ENABLE_WEB_MANAGEMENT=1
WEB_MANAGEMENT_PORT=8080
```

This will use the server machine as a time source, communicating over the network using NTP as a primary source, with a PTP backup. Both the server and the client will start TimeKeeper's internal web service, configured here on port 8080.

Next start/restart timekeeper to utilize the new configuration

```
$ systemctl restart timekeeper
```

> Non systemd-based systems can be started similarly with *service timekeeper restart*.

Verify it's working

From the command line you can run *tkstatus* (*tkstatus.bat* on Windows) to query the status of TimeKeeper and verify that it is working properly. For untruncated status and additional information, including one-way delay, you can run *tkstatus -f* (*tkstatus.bat -f* on Windows).

The web tools will be running on both systems, allowing you to view information about the configuration, timing quality and more with your web browser pointed to each machine on the configured port.

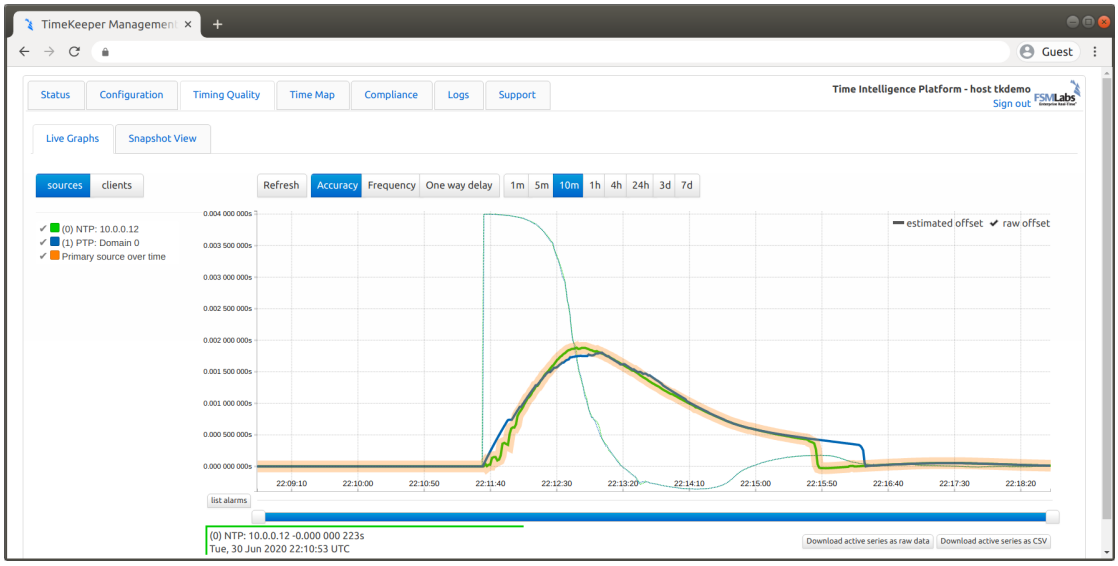
You can also review */var/log/timekeeper_0.data* and */var/log/timekeeper_1.data* for a log of updates. On Windows, the *var\log* path is in the same location as the TimeKeeper installation. The first column is the time since midnight 1970 January 1st of each update. The 2nd column is the offset from the remote server in seconds. More detailed information about each column is located in the "File formats" section.

Seeing a correction

In production, no other process should drive the clock other than timekeeper. For demonstration purposes, we will adjust the clock on the client and verify it is corrected by timekeeper. As root, on the client run:

```
$ /opt/timekeeper/release64/baddate .004
```

Since our client is tracking the server SOURCE0 and we abruptly changed the clock on the client, we should expect to see an abrupt change in time followed by a smooth slewing of the clock back to a time roughly consistent with the source. To verify this expectation we'll look at the TimeKeeper web interface under the Timing Quality tab:



Troubleshooting

Firewalls and blocked ports

Make sure that you have NTP (port 123) and PTP messages (ports 319 and 320) allowed through the firewalls on each system.

Licensing limitations

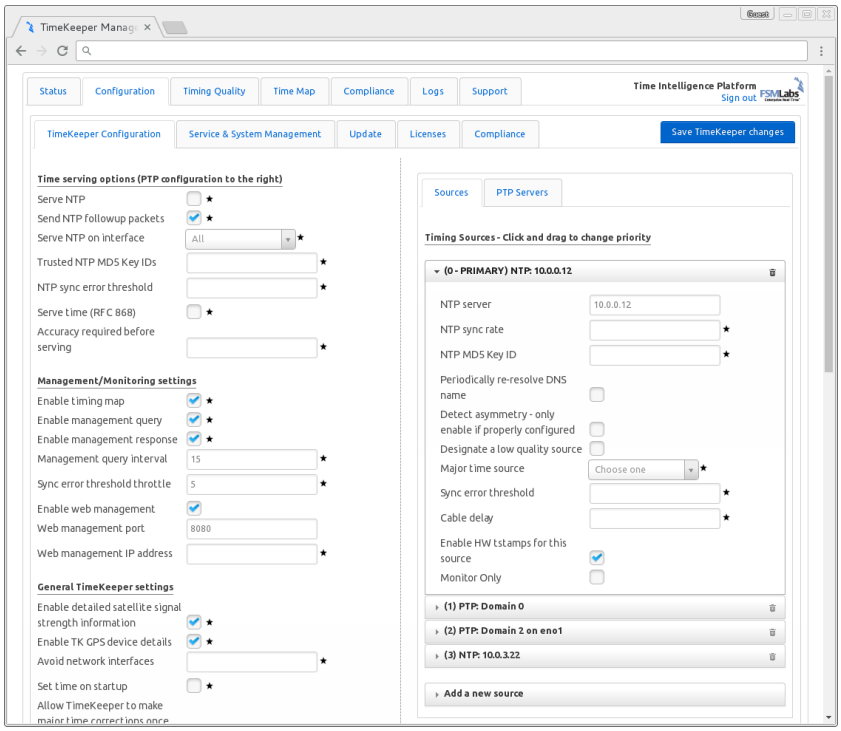
If you're running an evaluation version of TimeKeeper, it may only operate for a limited number of days or weeks before expiring. It may also limit the number of clients it can serve time to. These are evaluation limitations only

- if they prevent proper testing of TimeKeeper in your environment, please contact support@fsmlabs.com.

Common platform, common tools

Whether deployed as a hardware appliance or a timing client, TimeKeeper provides a consistent built-in web interface that allows users to configure any of the features described in this document. The configuration file can also be modified directly if desired. In the interest of being concise, examples will be given in the form of configuration file fragments.

We already saw the configuration file in the installation steps - for reference, here's a quick sample view of the configuration tools available via the web interface:



Swiss army knife of timing

TimeKeeper's abilities come from its core focus on tracking time very accurately from many different sources, and serving it out just as accurately, potentially in many ways.

This means TimeKeeper acts as a *Client* to start, in order to track all of these timing sources. It steers the local system time to these sources, accurate to the low nanoseconds, making use of any local hardware features it can find.

It then may act as a *Server* to put that accurate time back out on the network or other means to deliver it to downstream clients.

When these features come together in a hardware appliance, it's called a *Grandmaster*. We'll step through these feature sets and see how they build on each other, providing a common toolset along the way, from individual hosts running client software all the way up to redundant GPS and Rubidium-backed hardware Grandmasters.

First up, let's look at TimeKeeper as a timing client.

TimeKeeper Client

Use PTP, NTP, PPS, GPS - simultaneously

TimeKeeper can track time from just about any source. It can track PTP from a Grandmaster on the network, at the same time it tracks NTP from several different stratum servers, while also using a PPS and/or GPS input. This is useful to ensure proper delivery of timing data through the network - detecting network asymmetries, noisy clocks, faulty hardware, and more in addition to providing local clock accuracy in the nanosecond range.

Terminology

As mentioned above, TimeKeeper tracks ‘time sources’, which are defined in the configuration. We’ll cover a few common examples here to show how these sources work, with more specific detail as we go.

You’ll see the words ‘hardware timestamping’ throughout these documents. That refers to the ability of some network cards to more accurately timestamp packets as they’re received and sent. TimeKeeper will automatically make use of any of these features for a more accurate sync if they’re found. This allows for a more accurate and stable NTP and PTP sync, often in the low hundreds of nanoseconds.

NTP client

TimeKeeper supports nearly any form of NTP, and can achieve sub-microsecond accuracy with the protocol, which is a surprise to many. (NTP as a protocol is accurate, but many implementations of the protocol are not.)

As a client, TimeKeeper supports NTP versions 1, 2, 3, and 4, and can track NTP from any type of server, with accuracy primarily dependent on the quality of the NTP implementation on the server. (Generally, to achieve accuracy in the range of hundreds of nanoseconds, the NTP server providing time should be running TimeKeeper.)

A basic example of an NTP source, configured in TimeKeeper’s /etc/timekeeper.conf, is:

```
SOURCE0() { NTPSERVER=hostname; }
```

This will track NTP from ‘hostname’ as the primary source of time. There are a number of NTP options that can be enabled, which we’ll cover later - however, this is all that is needed to configure TimeKeeper for NTP.

PTP client

Similar to NTP, TimeKeeper will track IEEE 1588 PTP Grandmasters and boundary clocks from nearly any vendor, whether it’s a hardware appliance, a software server or a PTP-aware switch. TimeKeeper supports both PTP version 1 and version 2. Key features when using TimeKeeper as a PTP client include:

- Support for the default, telecom, and hybrid PTP profiles
- Ability to track many Grandmasters via many domains/interfaces simultaneously for controlled failover. This is in addition to the limited scope of the PTP BMC algorithm
- Ability to match network speed properly to avoid silent biases caused by network asymmetries

Below is a straightforward example that will listen for PTP domain 0 (the default) using the default profile as source 0, the default time source. This builds on our NTP example by keeping our NTP source, but demoting it to a secondary priority (source 1) for failover:

```
SOURCE0() { PTPCLIENTVERSION=2; }
SOURCE1() { NTPSERVER=hostname; }
```

PTP is very reliant on multicast traffic, particularly in the default profile. In this mode, time is distributed from the Grandmaster via multicast, and every client that communicates with the Grandmaster uses multicast. As you might expect, this does not always scale well - with thousands of nodes, every

client seeing every other client's Grandmaster request consumes a lot of network and processor bandwidth.

A hybrid mode often makes more sense, where the Grandmaster distributes time via multicast to clients, and the clients that need data (like delay information) from the Grandmaster make unicast queries directly to the Grandmaster. Again, let's build on our example, by adding a new source, tracking PTP domain 1 as the primary source, with our previous PTP source as the secondary source, and our NTP source as tertiary:

```
SOURCE0() { PTPCLIENTVERSION=2; UNICAST=1; PTPDOMAIN=1; }
SOURCE1() { PTPCLIENTVERSION=2; }
SOURCE2() { NTPSERVER=hostname; }
```

Since a hybrid PTP source is often used to avoid creating extra multicast traffic, normally if theres a hybrid source like source 0, a fully multicast source like source 1 would be removed. However, this is just is just an example so it's left for comparison.

Full unicast (telecom profile) is also an option. In this case, the client subscribes to unicast updates from the Grandmaster and communicates with the Grandmaster entirely via unicast traffic. This can be ideal for longer links where multicast is not an option. Building on our previous example, now with our primary source being a fully unicast PTP feed available on PTP domain 2, with fallbacks to a hybrid PTP source, a fully multicast PTP source, and then NTP:

```
SOURCE0() {
 PTPCLIENTVERSION=2;
 PTPSERVER=gm_hostname;
 PTPDOMAIN=2;
}
SOURCE1() { PTPCLIENTVERSION=2; UNICAST=1; PTPDOMAIN=1; }
SOURCE2() { PTPCLIENTVERSION=2; }
SOURCE3() { NTPSERVER=hostname; }
```

PTP is a fairly complex protocol, and many other options are available to finely tune your deployment. We'll cover other optional capabilities later in this document.

TimeKeeper communicates with just about any PTP Grandmaster, boundary clock, and client on the market. This includes but isn't limited to TimeKeeper Grandmasters, Symmetricom/Microsemi products such as the S3XX line (like the S350), SSU2000 models, TPXXXX models (like the TP5000), Spectracom hardware like the SecureSync and VelaSync, and also hardware from EndRun, Oscilloquartz, Arista, Cisco, and so on. If you have any specific questions about your hardware, please contact us at support@fsmlabs.com.

PCIe cards

A PCIe card can also be used to deliver time to a TimeKeeper client. The specific method will vary by the card's capabilities, but may be a direct GPS feed, an IRIG-B distribution, PPS, and so on.

> Note: In addition to these examples, some TimeKeeper users get time from their custom in-house hardware.

TimeKeeper can generally use the card in any supported mode, allowing the card to be configured as needed. Local system time can be driven based on the card's time and then redistributed, whether that's via one of the card's outputs or with TimeKeeper serving PTP and NTP on the network.

Out of the box support is provided for a number of cards from Spectracom, Symmetricom, and others.

In these cases, the card can be configured as needed to select timing inputs with the vendor supplied tools. Once configured, TimeKeeper will track the time provided by the card, along with any other configured sources.

Spectracom PCIe TSync GPS/IRIG Timecode Processor

A driver for this card is included with TimeKeeper and is loaded automatically when the Spectracom TSync card is selected as a source. You may also load and configure a driver for this device from the vendor. TimeKeeper does not change the reference table stored in the card so the reference table must be set up before starting TimeKeeper.

Adding a Spectracom TSync PCIe card to our existing example at the end of the PTP section would look like the following:

```
SOURCE0() { PPSDEV=spectracom; }
SOURCE1() {
 PTPCLIENTVERSION=2;
 PTPSERVER=gm_hostname;
 PTPDOMAIN=2;
}
SOURCE2() { PTPCLIENTVERSION=2; UNICAST=1; PTPDOMAIN=1; }
SOURCE3() { PTPCLIENTVERSION=2; }
SOURCE4() { NTPSERVER=hostname; }
```

In this case, the local card would be used first, with automatic failover to the PTP profiles and to NTP. Of course, even without failover, TimeKeeper will provide clear per-source timing details allowing you to cross check and alarm based on the behavior of all of your timing sources.

When using a PPS input the MAJORTIME option should be used to combine the PPS with time of day information from another source. More details on how to do this can be found in the section, “Major/minor sources”.

Symmetricom/Microsemi BC635PCIe/BC637PCIe IRIG, GPS and PPS cards

Similarly, a Symmetricom/Microsemi BC635/BC637 PCIe card could be used as a local source.

TimeKeeper can use these cards to receive a PPS, GPS or IRIG time references, depending on how the card has been configured. Only one card can be used at a time, and as with the Spectracom card, if only a PPS is delivered, the source should be combined with another source that provides time of day information, detailed in the “Major/minor sources” section.

Here, we’ve configured it to be the last source to be used, if all of the others fail, just as an example:

```
SOURCE0() {
 PTPCLIENTVERSION=2;
 PTPSERVER=gm_hostname;
 PTPDOMAIN=2;
}
SOURCE1() { PTPCLIENTVERSION=2; UNICAST=1; PTPDOMAIN=1; }
SOURCE2() { PTPCLIENTVERSION=2; }
SOURCE3() { NTPSERVER=hostname; }
SOURCE4() { PPSDEV=symmetricom; }
SYMMETRICOM_DRIVER_PATH=/opt/bc637_driver_release
```

In this configuration the drivers for the BC635/BC637 card must be installed separately from TimeKeeper and configured as desired with the driver’s tools. The driver’s location must be named using the SYMMETRICOM_DRIVER_PATH config option. This is the path to directory containing the subdirectories, dvrv/, samples/, etc.

EndRun GPS/Network/CDMA time reference device

TimeKeeper can use EndRun devices as a time source. The time source device should be configured to produce a NMEA ZDA string on the RS-232 port at 4800 baud. Using the FSMLabs provided cable, connect the BNC connector to the “1 PPS” output of the EndRun device and the RS-232 connector to the male DB9 port on the EndRun device labeled “Serial Time”. You will need a null modem adapter between the supplied cable and the “Serial Time” port.

Below is an example configuration using the reference input connected to the first serial device on the system, where the device is the last priority

source in the system. You can also use any other serial port on the system as long as you update “PPSDEV” to reflect that device. You may not use a USB to RS232 converter for this connection since they introduce a great deal of delay and jitter on the connection which will result in poor accuracy.

```
SOURCE0() {
 PTPCLIENTVERSION=2;
 PTPSERVER=gm_hostname;
 PTPDOMAIN=2;
}
SOURCE1() { PTPCLIENTVERSION=2; UNICAST=1; PTPDOMAIN=1; }
SOURCE2() { PTPCLIENTVERSION=2; }
SOURCE3() { NTPSERVER=hostname; }
SOURCE4() { PPSDEV=/dev/ttyS0; }
```

GPS, PPS tracking

One of TimeKeeper’s strengths is that it can consume data from the network and a direct GPS or PPS input or output. A sole PPS input can be combined with another source for time of day information (like a NTP or PTP source). Benefits of GPS/PPS inputs include:

- Highly accurate reference inputs for serving time or providing cross check
- Low cost - sub-microsecond accuracy is possible with just a PPS via a serial input

A PPS can be easily pulled into a system, whether it’s for an accurate time source directly or as a cross check for other sources. In some cases it’s provided via a PCIe card, described above. It can also be delivered directly to an FSMLabs NetCard device or via a serial line input.

As with the other scenarios, informing TimeKeeper of a source is simple. If a PPS is brought in on the serial port, TimeKeeper can combine it with another source that delivers time of day information (major time) even if the source of major time is relatively noisy. The result will be a very accurate clock based on the delivery of the second boundary from the PPS.

```
SOURCE0() {
 PPSDEV=/dev/ttyS0;
 MAJORTIME=SOURCE3;
}
SOURCE1() {
 PTPCLIENTVERSION=2;
 PTPSERVER=gm_hostname;
 PTPDOMAIN=2;
}
SOURCE2() { PTPCLIENTVERSION=2; }
SOURCE3() { NTPSERVER=hostname; }
```

Hyper-V support

On a virtualized host time may be available from the hypervisor directly. This is the case in some Hyper-V environments like cloud systems hosted in Azure.

To configure for Hyper-V, a source should be added using the PPSDEV option. Then just like in the other examples, additional sources can identify PTP and NTP servers for comparison/reporting/etc.

```
SOURCE0() { PPSDEV=hyperv; }
SOURCE1() {
 PTPCLIENTVERSION=2;
 PTPSERVER=gm_hostname;
 PTPDOMAIN=2;
}
SOURCE2() { NTPSERVER=hostname; }
```

Other sources can also be configured, as in the other examples. Here the primary source is the Hyper-V time, with failover/cross check/compliance

reporting sources over both PTP and NTP.

This option is supported on Linux systems hosted in a Hyper-V environment.

Failover and Sourcecheck

In all of the above scenarios, TimeKeeper uses sources in the order configured. Meaning, the system clock will be driven to match source 0 (or the lowest numbered source) as long as that source is delivering timing data. If source 0 stops providing data, TimeKeeper will fail over to source 1, 2, and so on.

If there's a source failure and TimeKeeper begins using source 1, it will continue to use source 1 until either source 1 fails or source 0 begins responding again. If source 0 returns TimeKeeper will begin tracking it again. All of this occurs regardless of what protocol or source is in use - NTP fails over to PTP, PTP can fail over to PPS, and so on.

TimeKeeper can also actively compare time sources and proactively reject bad ones using the Sourcecheck feature. This feature applies to TimeKeeper clients, servers, and Grandmasters. For details on enabling and using Sourcecheck, please refer to the "Sourcecheck" section.

TimeKeeper Server

We just saw how TimeKeeper acts as a *client* (see the "TimeKeeper Client" section), consuming time from a number of different sources. TimeKeeper server builds on this to take this time and use it to serve accurate time to clients.

As mentioned before, all of the capabilities described below can be managed easily with the built-in web interface. Configuration samples will be used here to keep the descriptions concise.

Given time sources to track, TimeKeeper can deliver PTP, NTP and TIME, and more to thousands of clients. Below we'll build on the previous NTP client snippet, so that the below configurations are standalone examples of how to serve time.

Serving NTP

NTP is a very accurate protocol that is often poorly implemented. Do not believe outlandish claims that NTP "cannot achieve accuracy better than a millisecond", it's simply not true. You're most likely to hear this from someone trying to sell you a PTP solution, or someone accustomed to poor NTP implementations. TimeKeeper serving NTP to TimeKeeper clients delivers accuracy much better than a microsecond without complex and expensive investments in specialized protocols and networking hardware.

NTP server benefits include:

- Accuracy in the low hundreds of nanoseconds with off the shelf hardware
- Management capabilities to record the reported accuracy of NTP clients
- Stratum capability - put TimeKeeper in front of your existing noisy NTP servers to deliver a smoothed and accurate time for your clients
- Support tens of thousands of clients per second from a single instance

Enabling the NTP server is a matter of adding this to the configuration:

```
SOURCE0() { NTPSERVER=hostname; }
SERVENTP=1
```

If TIME (RFC 868) is also desired, it can be enabled by adding "SERVETIME=1" to the above configuration.

NTP can be restricted to a specific interface if needed in a particular environment. It can also provide authenticated time, alert on client accuracy issues, and more.

PTP Grandmaster/boundary clock

TimeKeeper serves IEEE 1588 versions 1 and 2 in a number of ways. Since most users are interested in version 2 the examples focus on that.

Some terminology is in order here. A Grandmaster is generally a clock at the root of a timing network, delivering time to downstream clients, which may in turn deliver time to their clients.

A boundary clock is a PTP clock that takes time in from some upstream time source, and delivers time based on that source to any downstream clients. Some boundary clocks also serve as a bottleneck, preventing downstream clients from being able to reach upstream time sources. TimeKeeper does not add that limitation.

When serving PTP, TimeKeeper serves multiple domains via multiple interfaces, allowing PTP traffic to be matched properly with the network speeds. This avoids the introduction of link asymmetries that can cause clients to be unintentionally biased. This also allows for careful and complete redundancy, far beyond the limited failover capabilities of the PTP protocol itself.

Some key features of TimeKeeper as a PTP Grandmaster or boundary clock include:

- Support for the PTP default, telecom, and hybrid profiles

- Serves via many domains on many interfaces for link redundancy and managed PTP distribution
- Supports standard and extended management formats to collect client and Grandmaster information in one centralized location
- Can act solely as a management node on the network, collecting and analyzing timing data of discovered clients and Grandmasters

PTP (version 2) provides time in multiple 'domains', which can have different parameters and behavior. Clients can then track these individual domains as needed, for redundancy and comparison for example.

To serve PTP domain 0 (the default), just add this to our sample NTP server configuration:

```
SOURCE0() { NTPSERVER=hostname; }
SERVENTP=1
SERVEPTP0() { PTPSERVERVERSION=2; }
```

Note that TimeKeeper also supports PTP version 1. Configuring to use IEEE 1588 version 1 means that there would be a "1" in place of the "2" for the PTPSERVERVERSION above.

To also serve PTP domain 2, the configuration would be:

```
SOURCE0() { NTPSERVER=hostname; }
SERVENTP=1
SERVEPTP0() { PTPSERVERVERSION=2; }
SERVEPTP1() { PTPSERVERVERSION=2; PTPDOMAIN=2; }
```

In many cases, it's useful to serve PTP via specific interfaces. This ensures, for example, that PTP for a 10 gigabit network is being specifically delivered on a 10 gigabit link. It also provides clear redundancy and failover options for clients.

As an example, let's take the previous configuration, and serve the PTP domain 0 traffic via eth1, and PTP domain 2 via eth2.

```
SOURCE0() { NTPSERVER=hostname; }
SERVENTP=1
SERVEPTP0() { PTPSERVERVERSION=2; IFACE=eth1; }
SERVEPTP1() { PTPSERVERVERSION=2; PTPDOMAIN=2; IFACE=eth2; }
```

As with the client PTP configuration, there are more options and capabilities than covered here in this overview. The rest of the documentation covers the other options and benefits in more detail.

Synchronizing and serving time

At this point we can put together a single example of TimeKeeper tracking time from multiple sources and serving that time to clients. Below, TimeKeeper is tracking PTP on domain 0, with a fallback to unicast PTP (domain 1, via eth4) and then NTP. It will synchronize the local clock to these sources and serve it via PTP (domain 3) and any NTP clients.

```
SOURCE0() { PTPCLIENTVERSION=2; PTPDOMAIN=0; }
SOURCE1() { PTPCLIENTVERSION=2; PTPSERVER=grandmaster; PTPDOMAIN=1; IFACE=eth4; }
SOURCE2() { NTPSERVER=hostname; }
SERVEPTP0() { PTPSERVERVERSION=2; PTPDOMAIN=3; }
SERVENTP=1
```

In this case, TimeKeeper is acting as a boundary clock for PTP and a stratum server for NTP at the same time. NTP clients can use this server, and PTP clients can communicate with it with the default, hybrid, or telecom profiles. Any TimeKeeper clients could track both NTP and PTP from this system at the same time for protocol redundancy.

Serving time via PPS

TimeKeeper can also be configured to drive a PPS on the network to deliver accurate information for downstream clients. This allows for:

- A PPS input to another server for an accurate input, creating a low cost and accurate stratum server
- A single PPS can offer accurate timing to your clients directly
- Cross check and detect network biases, latency issues, and network asymmetries

An example of this is where a TimeKeeper server tracks time and then uses those sources to steer a PPS that is fed directly to downstream clients. The clients can then consume and compare that direct PPS feed to their NTP and PTP data, to confirm any biases or asymmetries in the network.

Next steps

Now we've seen TimeKeeper take in time from multiple sources and use them to serve time in various forms to clients. TimeKeeper can be run directly by users on their own hardware, managed right along with all of their other software with common tools as part of their estate.

Alternatively, these features are also packaged up in appliance form, known as a Grandmaster. Let's look at what TimeKeeper can do as an appliance.

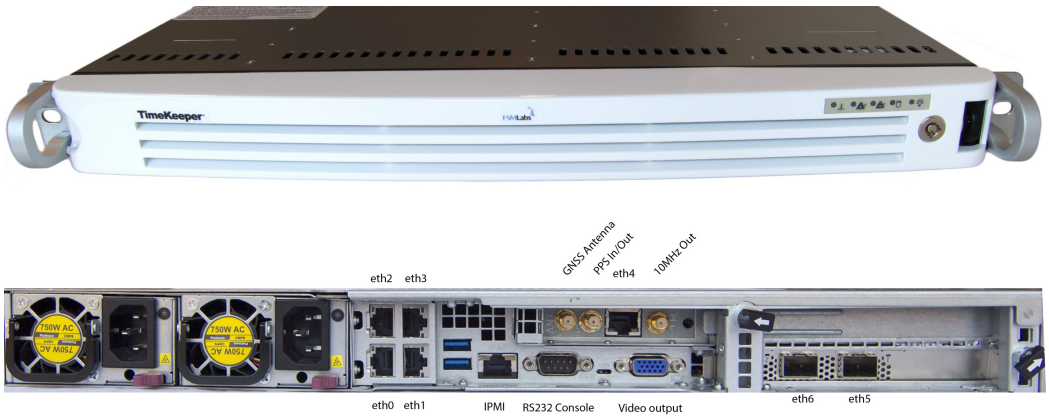
TimeKeeper Grandmaster

Now that you've seen how TimeKeeper acts as a *client* (see the "TimeKeeper Client" section) and then serves that time as a *server* (see the "Timekeeper Server" section), we can now look at how these features come together in the TimeKeeper Grandmaster Generation 2 appliance.

An appliance integrates TimeKeeper's features on a tested platform, and adds a number of features to provide a simple out of the box solution for users. These added features include:

- Integrated GPS receiver with support for multiple constellations
- High quality oscillator with 4us/day holdover, or a Rubidium option for 0.6us/day holdover
- Multiple 1G network interfaces
- Multiple 10G/25G network interfaces
- Reference PPS input
- Redundant storage (RAID)
- Redundant power supplies
- IPMI interface for out of band management and control
- Integrated PPS and 10MHz outputs

The Grandmaster uses a standard 1U platform:



Having an integrated device also allows for more capable software tools, including:

- SNMP query support
- User authentication via RADIUS or TACACS+
- Web interface via both HTTP and HTTPS
- Support for user-configured VLANs
- Support for user-configured bonded network devices
- Advanced routing capabilities
- Field upgradeable - get software improvements and updates over time

For more details on Grandmaster features, please refer to the section, "TimeKeeper Grandmasters."

Datasheets are also available at http://www.fsmlabs.com/.

VelaSync model

The FSMLabs Grandmaster is a drop-in replacement for many other Grandmaster models, including the Spectracom VelaSync. Note that FSMLabs no longer allows Spectracom/Orolia to manufacture TimeKeeper based Grandmaster appliances and is instead providing sales and support for VelaSync systems directly and via its resellers.

TimeKeeper features and core capabilities

TimeKeeper was built to provide enterprise scale timing solutions, with advanced alerting capabilities, visualization tools, and a wide array of features designed to fit a variety of timing needs.

Examples include:

- SNMP, syslog, email alerting for a wide range of timing events
- A built in web interface for visualization, configuration, and management
- Ability to combine time sources like a local high quality PPS and a low quality NTP server (major/minor sources)
- A timing map, automatically generated to give a high level view of the timing network
- Network asymmetry detection
- Ability to deliver local time as authoritative when there is no external reference signal
- Advanced source validation and rejection (Sourcecheck)

In this section, we'll look at each of these features in turn.

Alerting - Log files, SNMP, syslog, email, Windows event log

TimeKeeper provides several methods of delivering notifications of significant events. These notifications are called ‘alerts’. These are always logged in the main TimeKeeper log file (*/var/log/timekeeper* on UNIX systems and timekeeper.log on Windows). On Linux they can optionally be delivered via email, syslog, SNMP, or a combination of all three. On Windows TimeKeeper sends event information to the Windows Event log and optionally via SNMP. Below is an overview of the different types of alerts and each type of delivery mechanism.

Alert messages

The following sections contain the specific messages that TimeKeeper will emit via log files, event logs, SNMP, email, or syslog. Variable components of each message are replaced with the string VALUE below - depending on the specific value found, the message will vary here. Each alert is followed by a brief explanation of the cause, and are grouped by function.

Below these message are grouped by their SNMP trap type only to group them into functional categories. All of these messages are sent through all alerting mechanisms that TimeKeeper supports and not just SNMP traps.

clientQualityTrap

1. Client sync quality error on PTP server VALUE, client VALUE, source VALUE: absolute value of offset VALUE > VALUE
2. Client sync quality error on NTP server 0, client VALUE: absolute value of offset VALUE > VALUE
3. Client VALUE Source VALUE: inactive for VALUE seconds > VALUE seconds
4. Client VALUE Source VALUE: is now active again for VALUE seconds

TimeKeeper grandmasters collect client sync accuracy information from clients - and will emit the above messages when one of the clients exceeds the server-specified threshold. (See the per-PTP-server configuration variable SYNCERRORTHRESHOLD, the global NTP configuration variable NTPSYNCERRORTHRESHOLD and the INACTIVE_CLIENT_THRESHOLD global configuration variable.)

Messages 1 and 2 can be throttled by setting the *SYNC_ERROR_THRESHOLD_THROTTLE* variable. These messages can be limited to the primary source only by setting the *SYNC_ERROR_THRESHOLD_ALERT_ONLY_PRIMARY* variable.

Message 3 will only be sent if TimeKeeper has detected that a client has become inactive for the period of time specified by the INACTIVE_CLIENT_THRESHOLD configuration option.

Message 4 is a clearing trap for Message 3 - if TimeKeeper previously detected that a client was inactive and has since become active again, this message provides notification of the change.

When configured to send SNMP traps, this event will be delivered using the clientQualityTrap.

licenseStateTrap

1. License problems detected at startup. Please contact support@fsmlabs.com.
2. Unable to get license information for TimeKeeper VALUE, continuing for VALUE seconds. (status VALUE)
3. TimeKeeper VALUE: Had license status issues, but repeated checks now indicate correct state. Continuing.
4. Warning: TimeKeeper VALUE license pending expiry in VALUE days.
5. TimeKeeper VALUE license expired, in grace period, continuing for VALUE days.
6. Unable to get valid license information, due to expiry or repeated failed checks. TimeKeeper shutting down.

The above messages are sent when TimeKeeper needs to alert the user to licensing issues.

Message 1 is only sent on startup if TimeKeeper cannot get an initial license to use. There is no clearing trap for message 1 as TimeKeeper is not starting up.

Message 2 indicates TimeKeeper is unable to contact a license server (if in use) to get license data, and will continue for the specified time. Message 3 is a clearing trap for message 2. This will only ever occur if TimeKeeper gets its license from a remote site, was unable to reach that site for a period of time, and then regains connectivity.

Messages 4 and 5 indicate that either a license is going to expire, or has expired and is in a grace period, so a new license can be put in place before it becomes an issue. There are no clearing traps for these messages. Once a new license file is in place and active they won't be triggered.

Message 6 is sent when license checks have failed continuously and TimeKeeper is unable to continue. There is no clearing trap for this message.

When configured to send SNMP traps, this event will be delivered using the licenseStateTrap.

sourceStateTrap

1. Source VALUE: source appears to be valid again - re-enabling
2. Source VALUE declared invalid/insecure
3. Source VALUE: TimeKeeper GPS/Oscillator detected critical jamming level
4. Source VALUE: TimeKeeper GPS device reports overheating, Temp: VALUEC/VALUEF
5. Source VALUE: TimeKeeper GPS/Oscillator detected VALUE% jamming/noise
6. Source VALUE: u-blox GNSS receiver detected critical jamming level
7. Source VALUE: u-blox GNSS receiver detected VALUE% jamming/noise
8. Source VALUE: inactive for VALUE seconds > VALUE seconds
9. Source VALUE: PTP reported GM UTC/TAI offset change that appears incorrect VALUE -> VALUE
10. Source VALUE: post startup offset error > VALUE, setting to reference time
11. Source VALUE: TimeKeeper GNSS/Oscillator detecting sustained jamming, this may result in degraded holdover accuracy

If TimeKeeper is crosschecking sources with the SOURCECHECK feature, some sources may be found to be invalid and rejected for a time. If a source has been rejected but is now found to be valid again, message 1 above will be sent. This is a clearing trap for the remainder of the list.

Similarly, TimeKeeper will send trap 2 above when a source is rejected.

Messages 3-5 indicate there is an issue with the GPS device provided in TimeKeeper Grandmasters. These messages do not have clearing equivalents, and may indicate 'GNSS' rather than 'GPS' specifically.

Messages 6 and 7 indicate that there is an issue with a u-blox GNSS receiver.

Message 8 indicates that the source has stopped responding for a time that exceeds the outage time limit. The current outage limit is 180 seconds. This message is enabled by setting the *ENABLE_INACTIVE_SOURCE_ALERT* variable.

Message 9 can occur when the GM reports a change in the UTC/TAI offset that appears to be wrong or not possible.

Message 10 is provided when ALLOW_SET_TIME_AFTER_STARTUP is set and TimeKeeper is forced to set the time to correct for a large jump in time, like when recovering time from a suspended VM. This provides context when matching events to large offsets reported in TimeKeeper Compliance audits.

Message 11 indicates that TimeKeeper has determined that jamming to be an ongoing issue. Because of the this, oscillator training may not be complete. If the GNSS signal is lost in this state and the oscillator is used for holdover the oscillator accuracy may not be ideal. If jamming is expected to continue it's recommended that the HOLDOVER_LIMIT value for the source be reduced. This way if GNSS signal is lost comparisons against backup sources will begin sooner. When applicable, this alert is emitted roughly once per day.

When configured to send SNMP traps, this event will be delivered using the sourceStateTrap.

changedSourceTrap

1. Time source change VALUE -> VALUE (source VALUE -> source VALUE)
2. Unable to find any valid time source.
3. Previously unable to find any valid time source, now using source VALUE.

Message 1 above will be sent whenever TimeKeeper changes source for any reason - such as when one source stops providing timing data for more than 3 minutes. This will also occur in situations where the source TimeKeeper is tracking changes where it is getting time from, such as when an NTP

server switches to another upstream source.

Message 2 will only be sent if TimeKeeper has failed over to the last configured source and even that has stopped responding. At this point TimeKeeper is unable to track any sources and is driving the clock based on its best known clock rate.

Message 3 is a clearing trap for the second - if TimeKeeper previously lost all sources but one was found to be usable again, this message provides notification of the change.

When configured to send SNMP traps, this event will be delivered using the changedSourceTrap.

sourceQualityTrap

1. Sync quality error on source VALUE, absolute value of offset VALUE > VALUE
2. Sync quality restored on source VALUE, absolute value of offset VALUE < VALUE
3. Leap second will be inserted today

When a client has the SYNCERRORTHRESHOLD configuration option defined on a source, it will emit message 1 above whenever the sync offset exceeds that threshold. When the sync comes back into the SYNCERRORTHRESHOLD range, the message 2 will be sent as a means of clearing state.

When configured to send SNMP traps, this event will be delivered using the sourceQualityTrap.

For more information on leap second handling, see the "Leap seconds" section.

grandMasterFaultTrap

1. Grandmaster unable to collect system chassis version information.*
2. Grandmaster unable to collect system chassis information from pipe.*
3. Grandmaster power supply 1 failure
4. Grandmaster power supply 2 failure
5. RAID fault detected.*
6. Unable to confirm RAID status
7. Inlet temperature failure
8. Health check failed. *
9. S.M.A.R.T. attribute * has reached its pre-fail threshold. Possible imminent * drive failure.

The *s above indicates there may be further information provided in the data, such as which disk is failing in the case of the RAID events.

These messages are specific to TimeKeeper Grandmasters, and indicate issues with the appliance. Contact support@fsmlabs.com for assistance. These messages do not have clearing equivalents.

When configured to send SNMP traps, this event will be delivered using the grandMasterFaultTrap.

startupFaultTrap

1. Address VALUE specified for WEB_MANAGEMENT_IP but was not matched to a device. Ignoring parameter.
2. Cannot open socket: VALUE
3. Source VALUE: Cannot open socket: VALUE
4. Source VALUE: Unable to prepare server socket descriptors for PTP
5. Unable to prepare server socket descriptors for PTP
6. Invalid MAJORTIME reference specified for source
7. Invalid MAJORTIME SOURCE value specified on source VALUE
8. No time sources found. Make sure at least one time source is declared in /etc/timekeeper.conf
9. Invalid CPU value specified: VALUE
10. Error, LOGDIR specified but invalid: VALUE
11. Initial license check failed
12. Error processing PTP server argument VALUE, server VALUE

13. Error processing source argument VALUE, source VALUE
14. Unable to create Compliance directories
15. Unable to create Compliance directory VALUE
16. Unable to chown Compliance directory VALUE
17. Unable to look up nobody user
18. SNMPTRAPPASSPHRASE decode failed, ignoring parameter
19. SNMPTRAPPASSPHRASE was less than 8 characters long, ignoring parameter
20. WARNING! OS reports clock resolution (VALUE seconds) below what is required for accurate operation.
21. TYPE VALUE: Cannot open socket: IPv6 not supported.
22. Install is invalid. Please contact support@fsmlabs.com.
23. Source VALUE: Invalid interface specified: 'VALUE'
24. HTTPS_KEY_PASSPHRASE decode failed, ignoring parameter
25. External environment failed consistency checks, exiting. VALUE
26. Warning, invalid TLS_VERSION VALUE specified (must be VALUE). Ignoring parameter.
27. Cannot specify both IPV6 and PTP_LAYER2.

On startup, if the WEB_MANAGEMENT_IP parameter is used to limit access to the web tools, TimeKeeper makes sure that matches a configured interface. If it does not, message 1 will be sent out and TimeKeeper will allow the web interface to be used on all interfaces so the issue can be resolved.

If any source or PTP server has IPv6 enabled and IPv6 is not supported, message 21 will be sent out. 'TYPE' indicates the configuration type of 'Source' or PTP 'Server'.

The rest of the messages are a subset of the number of possible traps, sent on startup if TimeKeeper is unable to prepare internal resources as needed based on the current configuration, or if it's unable to start for other reasons, like an invalid configuration. TimeKeeper will be available for reconfiguration via the web interface (if enabled). TimeKeeper Grandmaster appliances always run the web interface so that reconfiguration is always possible.

When configured to send SNMP traps, this event will be delivered using the startupFaultTrap. Regardless of the specific message, a startupFaultTrap indicates TimeKeeper is not operating normally and requires investigation - generally this is due to a misconfiguration.

systemAlertTrap

1. Realtime clock changed by something other than TimeKeeper: expected ticks VALUE but was VALUE (adjtime VALUE)
2. Realtime clock changed by something other than TimeKeeper: expected freq VALUE but was VALUE (adjtime VALUE)
3. Realtime clock changed by something other than TimeKeeper: expected constant VALUE but was VALUE
4. Realtime clock changed by something other than TimeKeeper: expected offset 0 but was VALUE
5. Clock changed by something other than TimeKeeper: expected freq VALUE but was VALUE
6. Clock changed by something other than TimeKeeper: expected tick VALUE but was VALUE
7. Clock changed by something other than TimeKeeper: expected offset 0 but was VALUE
8. Clock changed by something other than TimeKeeper: expected adjustment VALUE but was VALUE
9. Active slave for VALUE changed from VALUE to VALUE

Messages 1-8 are sent when TimeKeeper suspects that something else is driving the system clock (see the "Clock adjustment and steering" section for details).

TimeKeeper sends message 9 when a different slave in a bond becomes active, suggesting that the previously active slave failed.

When configured to send SNMP traps, this event will be delivered using the systemAlertTrap.

Alerting Mechanisms

This table shows which alerting mechanisms are available on which platforms:

	Grandmaster	Linux	Solaris	Windows
TimeKeeper log	Y	Y	Y	Y
SNMP	Y	Y	Y	Y

	Grandmaster	Linux	Solaris	Windows
email	Y	Y	Y	
syslog	Y	Y	Y	
Windows event log				Y

SNMP

TimeKeeper can generate SNMP traps if configured in *timekeeper.conf* on Linux and Windows. TimeKeeper uses the program *snmptrap* from the package net-utils to send traps. This program is not included with TimeKeeper on Linux but is commonly included with Linux distributions. On Windows a version of *snmptrap* is included.

To configure TimeKeeper to generate SNMP traps and deliver them to the host *host1* add the following line to the *timekeeper.conf* file:

```
SNMPTRAPHOST=host1
```

host1 may be a hostname or an IPv4 dotted notation address. TimeKeeper supports multiple SNMP trap destinations and they can be listed as comma separate values:

```
SNMPTRAPHOST=host1,host2,host3
```

If only *SNMPTRAPHOST* is specified, TimeKeeper will use SNMPv2.

TimeKeeper also supports SNMPv3 User-Based Security Model and currently uses MD5 for authentication with DES for privacy. To configure TimeKeeper to use SNMPv3 traps, with authentication and privacy (authPriv security level), add the following lines to the *timekeeper.conf* file:

```
SNMPTRAPUSERNAME=DemoName
SNMPTRAPPASSPHRASE='encryptedpassphrase'
```

Note *SNMPTRAPPASSPHRASE* is used for both the authentication and privacy passphrases. Because it is stored in encrypted form, the value must be set using the TimeKeeper GUI, or by using the *encodepassphrase* utility provided with TimeKeeper.

If the parameter *SNMPTRAPEID* is set in *timekeeper.conf*, TimeKeeper will use this parameter to generate the engineID. If the parameter is not set, TimeKeeper will use the MAC address of the default network interface to generate the engineID.

If the *SNMPTRAPOID* parameter is set in *timekeeper.conf*, TimeKeeper will emit all traps to that OID. This is to retain legacy TimeKeeper behavior where all traps were by default sent to OID *1.3.6.1.2.1.16.0.1*. Unless you need all traps sent to one OID, it is recommended you leave *SNMPTRAPOID* undefined in your configuration.

SNMP MIB

By default, TimeKeeper delivers traps according to the provided MIB file, which can be found at:

> /opt/timekeeper/management/snmp/timekeeper_mib.txt

on Linux, and at:

> C:\Program Files\timekeeper\management\timekeeper_mib.txt

on Windows.

TimeKeeper Grandmasters also allow users to walk the SNMP tree on the appliance. To enable this feature, click *Enable SNMP queries* in the web interface, under *Configuration*, subtab *Service & System Management*. The SNMP tree can be walked remotely with standard tools - for example, assuming the Grandmaster is at IP 10.1.0.3, with SNMP community string 'public':

```
snmpwalk -v2c -c public 10.1.0.3 .1.3.6.1.4.1.42733.0
```

The SNMP values are cached over a short period in order to speed up SNMP queries in all conditions. Do not depend on querying SNMP at high rates to detect timing errors such as when a time source exceeds a named threshold. Instead, rely on TimeKeeper's ability to send SNMP traps to your management host as the event occurs. This provides immediate notification of errors and allows you to benefit from all of the different scenarios that TimeKeeper can detect.

The community string can be changed via the TimeKeeper web interface on the Grandmaster over HTTPS.

On TimeKeeper Grandmasters, in addition to the TimeKeeper-specific objects, you can query system objects via the following OIDs.

.1.3.6.1.4.1.2021.4	UCD-SNMP-MIB::memory
.1.3.6.1.4.1.2021.10	UCD-SNMP-MIB::laTable
.1.3.6.1.2.1.25.1	HOST-RESOURCES-MIB::hrSystem
.1.3.6.1.2.1.25.2	HOST-RESOURCES-MIB::hrStorage
.1.3.6.1.2.1.2	IF-MIB::interfaces
.1.3.6.1.2.1.31.1.1	IF-MIB::ifXTable

Walking SNMP tree on non-grandmasters

TimeKeeper does allow the SNMP tree to be walked on non-Grandmaster configurations, such as clients, servers, and boundary clocks. However, the specific configuration may vary depending on the distribution in use. A recipe that will work on many installations is to add these two lines to your *snmpd.conf*:

```
view systemview included .1.3.6.1.4.1.42733.0
pass .1.3.6.1.4.1.42733.0 /opt/timekeeper/management/snmp/snmphandler
```

An additional two lines are needed for SNMPv3 support :

```
createUser DemoName MD5 DemoPassphrase DES
rwuser DemoName priv
```

Please contact support@fsmlabs.com for more details if needed.

Syslog

Events that are sent over email or SNMP can also be delivered via syslog. TimeKeeper automatically emits syslog messages, so TimeKeeper clients can be configured to send data to a syslog server just like any other application, using whichever syslog daemon is in use on the client.

With TimeKeeper Grandmasters, the ability to configure where syslog data is sent is available in the web GUI. To change the configuration, log into the web interface, navigate to *Configuration* and then *Service & System Management*. Under *Manage Communication*, select *Configure Syslog*.

From this interface, 3 separate syslog servers can be configured. Once applied, the change is immediate and syslog messages will be delivered to those hosts without having to restart the system or TimeKeeper.

Email

TimeKeeper can generate emails for events if configured in */etc/timekeeper.conf* with the *EMAILNOTIFICATION* parameter. An example of this is:

```
EMAILNOTIFICATION=test@example.com
```

or, if multiple addresses should receive notification, use a comma-separated list:

```
EMAILNOTIFICATION=test@example.com,another@example.com
```

TimeKeeper uses the *mail* program to send the notifications. As with the *snmptrap* application, TimeKeeper does not provide the application but relies on the host configuration and network connectivity to deliver mail.

Emails may be throttled and bundled together using the *EMAILNOTIFICATION_THROTTLE* option, detailed in the "Global options" section.

Windows event log

As detailed above, on Windows TimeKeeper will automatically send information to the Windows event log (and optionally via SNMP). Any event that would be logged via SNMP, email, or syslog on Linux will be sent to the event log, so if sync error thresholds are configured and the threshold reached, that event will be logged locally at a minimum.

On the host itself, the details of these events can be viewed with the normal *Event Viewer* tool.

Logging configuration changes and events

On both TimeKeeper Grandmasters and software installations of TimeKeeper, notable updates made through the web interface (and `tkctl` on the Grandmasters) are logged to syslog on Linux and the event log on Windows. For environments that require change tracking this allows for centralized recording of changes made to all types of TimeKeeper installations.

Changes are reported as system/event log updates of the form:

```
TimeKeeper set successful: gm.config.timekeeper by user admin
```

This notes the specific change (TimeKeeper configuration change in this case), whether it was made (successful) or not (attempted), and the user who made the change. The event/system log notes the time and day it was done.

Changes that are logged are 'set' operations or change actions as defined by the `tkctl` tool. A list of possible named changes/actions are listed in the "tkctl options" section. Some options that apply to non-grandmaster installations, like updating the TimeKeeper configuration file, are also logged. For non-grandmaster installations the name will be prefixed with 'tk.' rather than 'gm.' as in the example above.

Events that are significant will be sent in the same way as changes, with the name 'gm.event.eventname' or 'tk.event.eventname' depending on whether it's on a grandmaster or a software client. For example, logins are recorded for success or failure like this:

```
TimeKeeper event successful: gm.event.login by user admin
```

on a TimeKeeper Grandmaster. On a software TimeKeeper installation the event name would be 'tk.event.login', and in both cases if the login failed 'successful' would be replaced with 'attempted.'

The following events are recorded in addition to the configuration changes reported above.

- gm.event.login/tk.event.login
- gm.event.basepackageinstall

- gm.event.timekeeperinstall
- gm.event.licenseupdate

Web management

Many of TimeKeeper's tools and abilities are shown through its web interface. In this section we'll work through how to enable and use this tool at a high level. For specific details on any features contained within the web interface, please refer to the specific section on that feature.

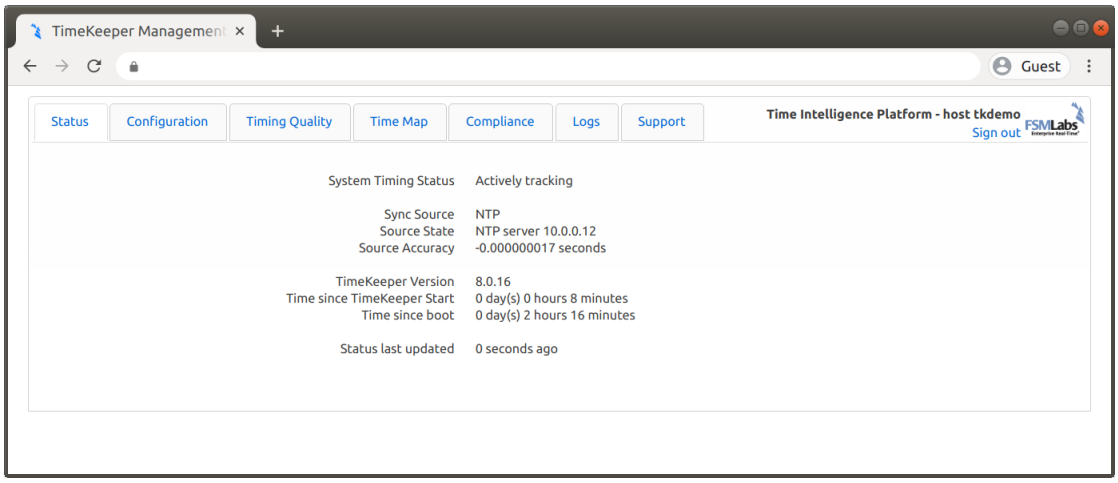
Enabling web management

If configured and licensed, TimeKeeper will offer set of management tools via a web server running with TimeKeeper. To enable this feature, the configuration file should include:

```
ENABLE_WEB_MANAGEMENT=1
WEB_MANAGEMENT_PORT=80 # to use the standard HTTP port 80
```

If *WEB_MANAGEMENT_PORT* is not specified in the configuration, but the *ENABLE_WEB_MANAGEMENT* option is set to 1, TimeKeeper will start the web management tools over HTTPS on port 443. Please see the web interface section for more details.

A web browser pointed to the identified management port (or 8080) will show a front page similar to:



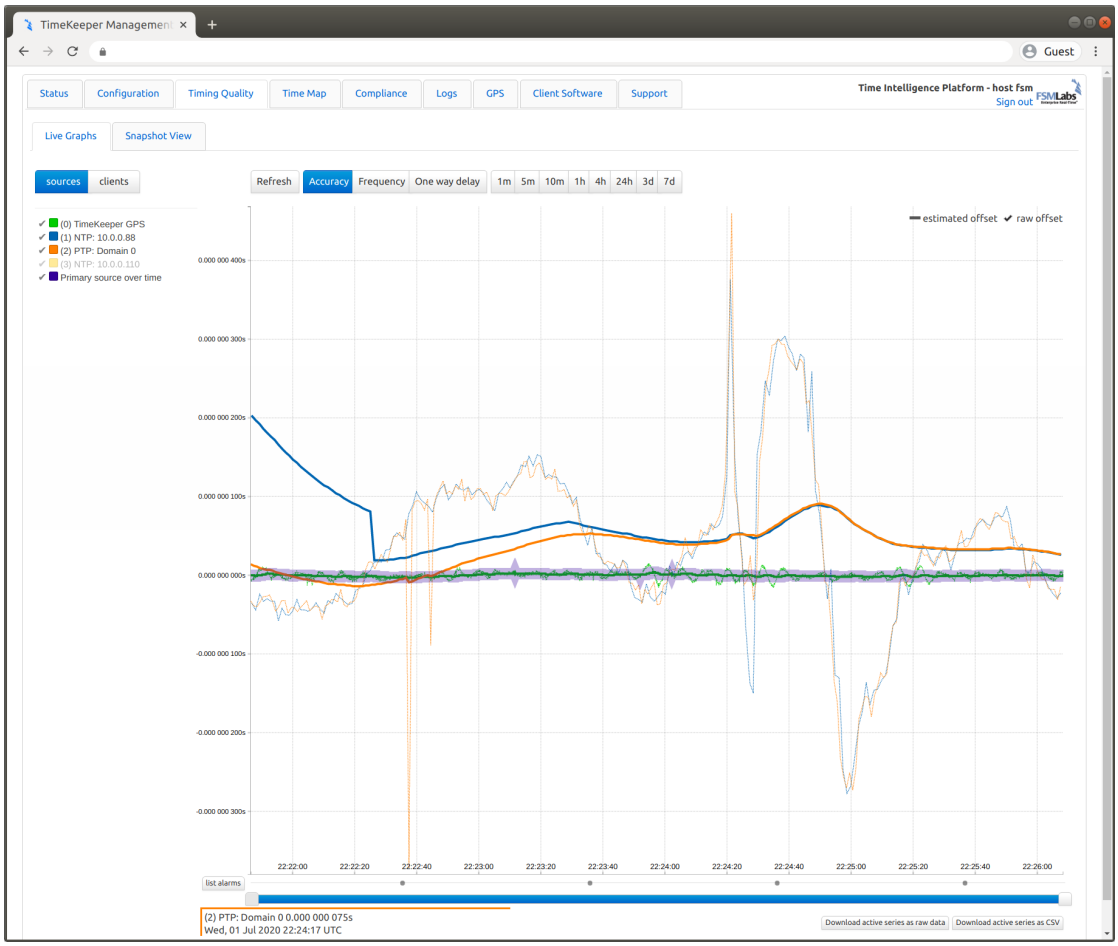
Refer to the "Authentication - users and protocols" section for login details. Chrome and Firefox are recommended. Mobile browsers can be used to connect to the administrative console. Should you have specific browser requirements, contact support@fsmlabs.com.

Via this interface, many TimeKeeper features are made visible - configuration, plotting of timing accuracy, tracking of any clients, a map of the timing network, and more.

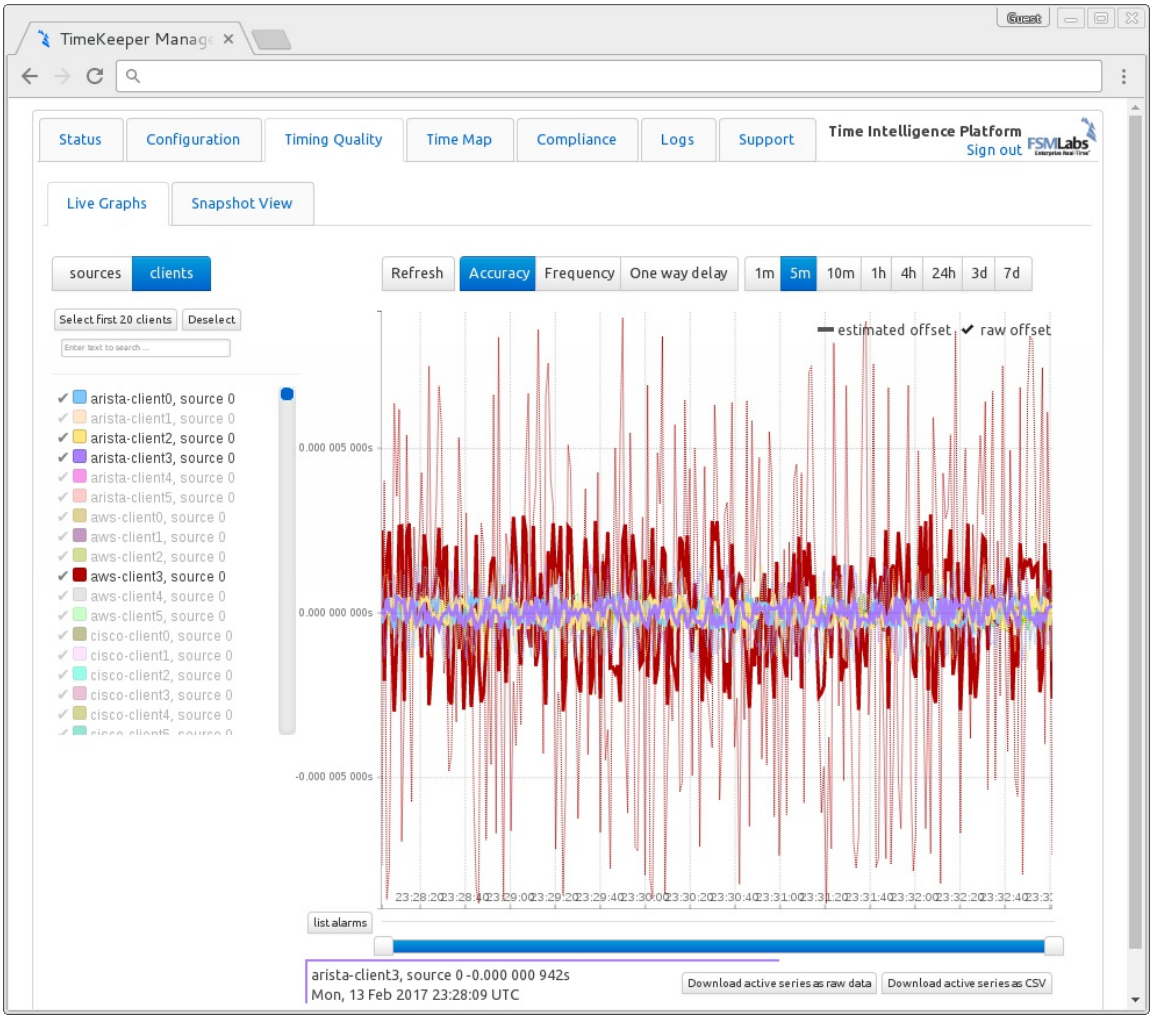
Timing accuracy visualization

The web interface allows you to see how all of your time sources are behaving compared to each other. This includes offset information, one way delay data, and more. The data is shown on the *Timing Quality* tab, under *Live Graphs*.

The plotted sources and quality will vary by deployment. Here is an example of TimeKeeper plotting timing accuracy against the configured sources. Note the primary source over time is highlighted:



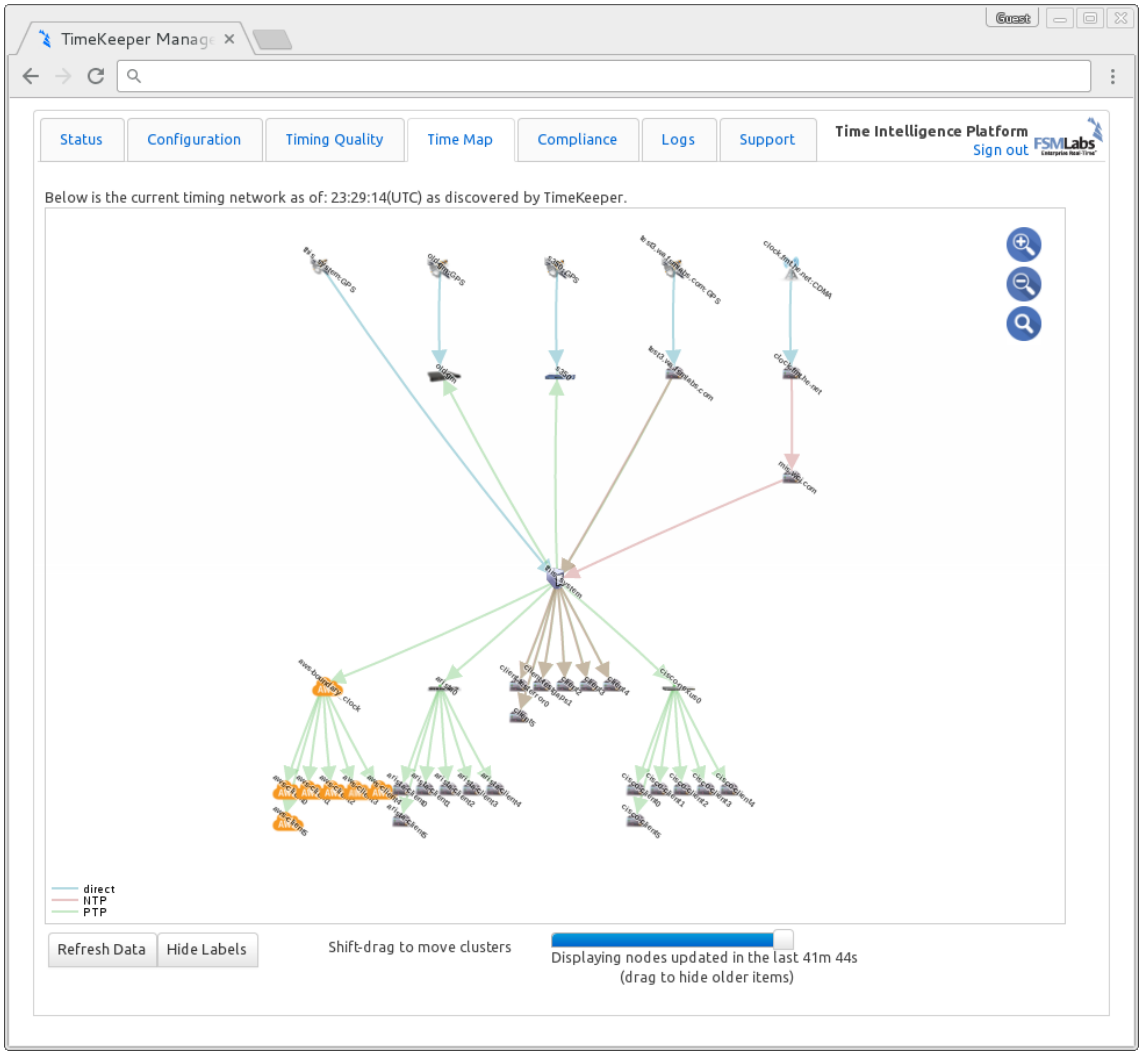
And here's an example showing the accuracy of some clients reporting their data to an instance of TimeKeeper:



Timing map

TimeKeeper also builds a map of the timing network, from your sources to their sources as far as can be reached. It also includes any other clients found on the network, and other time sources even if it's not a configured source on the local host. These maps can be very extensive and in some cases point out issues with the current timing systems in place.

Here is a snapshot of a map generated with TimeKeeper:



We'll cover what that map will display in more detail in the "Timing map and network view" section. Also, there are more details about the web interface in the "Visualizing data" section.

Sourcecheck

The TimeKeeper ‘Sourcecheck’ feature is an analytics and automated threat detection tool that allows TimeKeeper to detect bad time sources and switch away from them. Bad time can come from misconfiguration, equipment and software failures or intentional time based attacks. Sourcecheck can detect and defeat them.

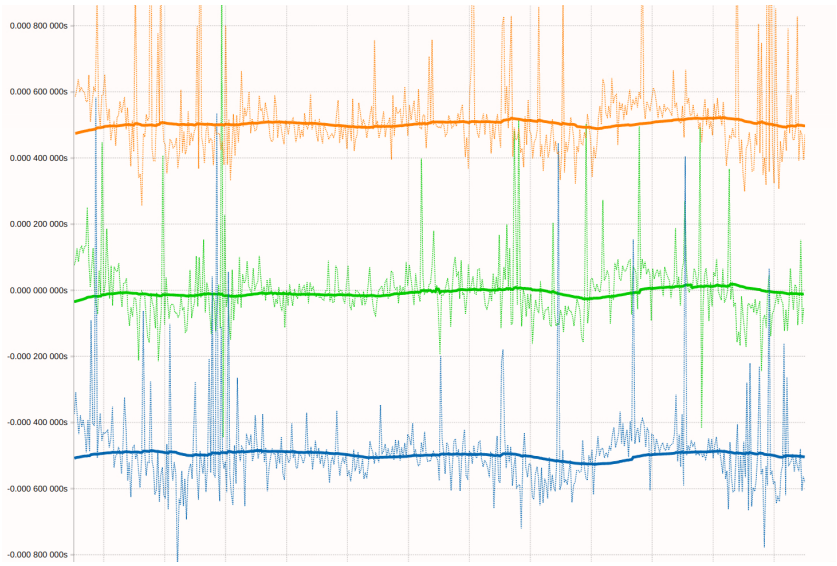
Sourcecheck overview

Sourcecheck is a configurable option in TimeKeeper and should only be selected if there are high quality alternate sources. When Sourcecheck is disabled, TimeKeeper will accept the highest priority configured source (NTP, PTP, GPS or other) as authoritative. TimeKeeper will apply sophisticated filtering to reduce jitter and it will failover to the next source and produce alerts if the highest priority source stops communicating new times (or violates protocol). If the highest priority source resumes operations later, TimeKeeper without Sourcecheck will revert. This behavior avoids a number of serious problems caused by ntp implementation attempts to second guess time and by the simplistic “Best Master Clock” operation of PTP 1588 while preserving interoperability. When Sourcecheck is enabled TimeKeeper will cross-check all the existing time sources with one another to validate them. If the highest priority time source fails the Sourcecheck validation TimeKeeper will mark it as invalid, generate alerts and switch to the next highest priority time source that is not currently marked invalid. Cross check detects bad time from equipment failures, misconfiguration, false leap seconds as well as malicious activities such as GPS spoofing and other time based attacks.

How it works

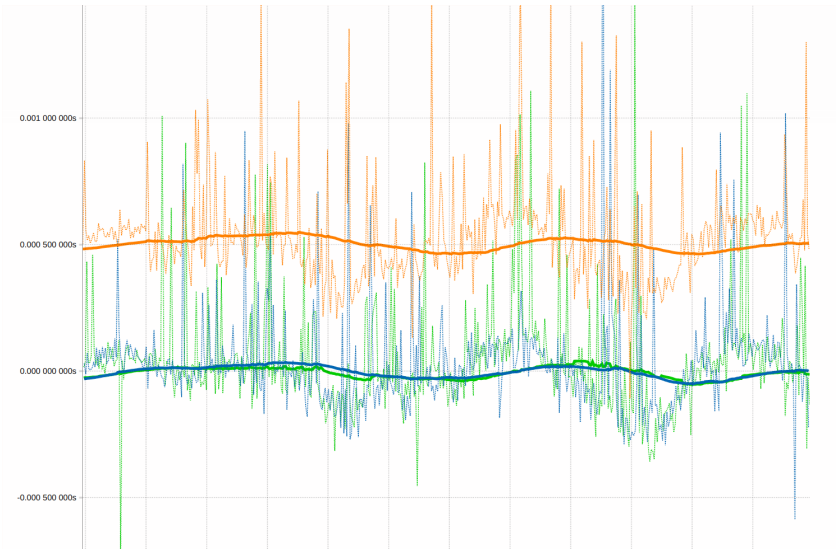
Sourcecheck combines a multi-source stochastic analysis, protocol verification, and source voting to detect outlier time sources. Each time source gets a vote for the correct time. TimeKeeper then looks for patterns over an interval to see if a second source is more in sync with a critical mass of sources.

Below you can see a simple example of a tie with 3 time sources. The primary (source 0, in green) and two secondary time sources all disagree. The secondary time sources disagree from the primary by +0.5 and -0.5 milliseconds respectively. Since there is no ‘quorum’ here source 0 remains valid. Note that for some examples below, the source disagreements are large for ease of illustration. In practice these discrepancies will generally be much smaller and Sourcecheck will be identifying issues with sources just a few microseconds apart.



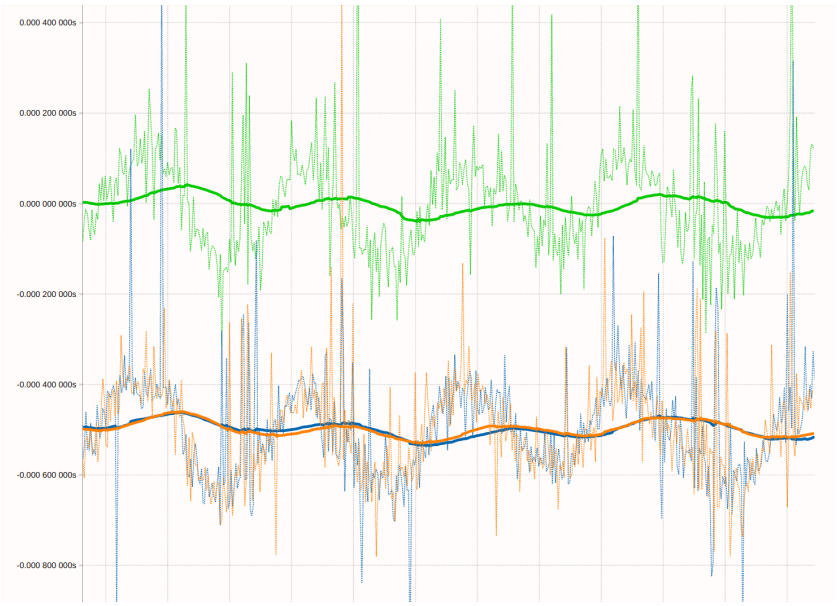
Source 0 (green) disagreeing with both source 1 (blue) and source 2 (orange).

Next you can see a chart showing that source 0 and 1 agree while source 2 is offset by -0.5 milliseconds. Sourcecheck will declare source 2 as invalid in this case.



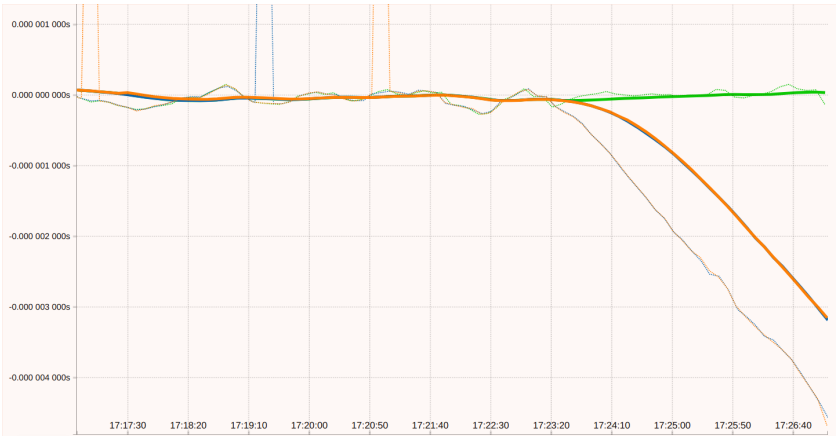
Source 0 (green) and source 1 (blue) agree, but source 2 (orange) is invalid.

Below you can see an example where source 0 would soon be declared invalid. That is because source 1 and 2 form a quorum, allowing source 0 to be rejected. Once Sourcecheck flags source 0, TimeKeeper would allow failover to source 1 so that the offset can be corrected for. Again please note the scales of disagreement here are for illustrative purposes, and in many cases Sourcecheck identifies similar issues but at a much smaller scale, down in the low microsecond range.



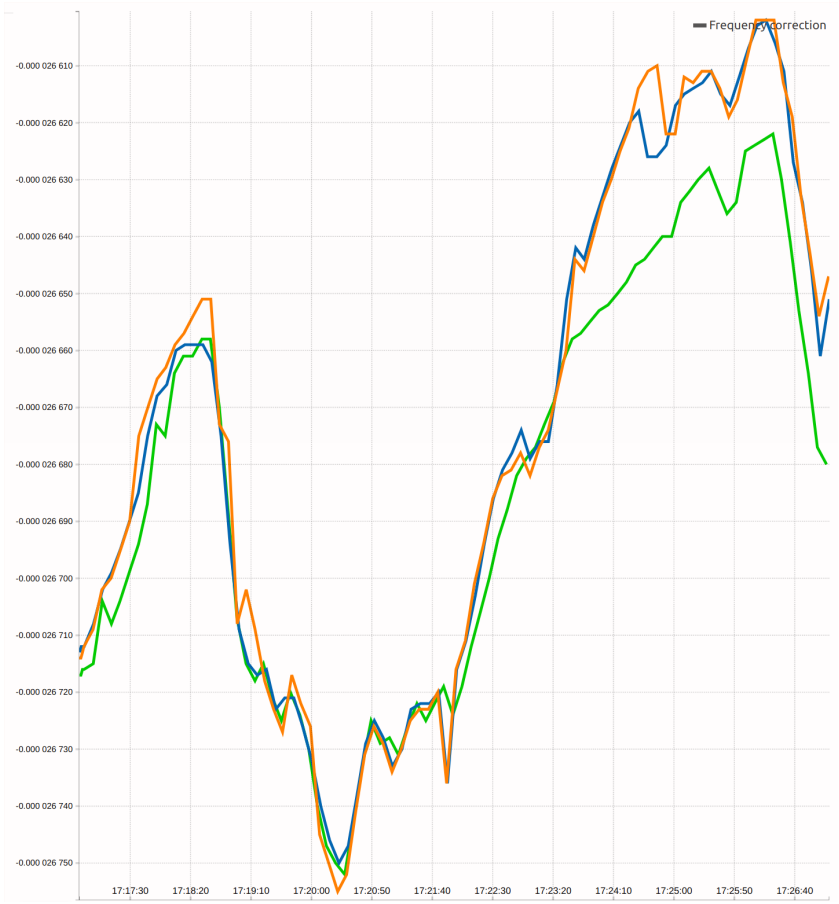
Source 0 (green) is invalid and will be rejected since source 1 (blue) and source 2 (orange) agree.

Sourcecheck also analyzes by both offset and frequency. In the example below without Sourcecheck taking action we see the secondary and tertiary time sources start slowly drifting in offset around 17:24 because a GPS spoof attack is under way. The attack is slowly moving the time on our primary time source (source 0) but in this graph it looks as though all the other time sources are moving because TimeKeeper is still tracking source 0. Eventually Sourcecheck's time offset analysis would catch the compromise but frequency analysis spots the problem earlier.



Source 0 (green) is being spoofed but it can be some time before source 1 (blue) and source 2 (orange) can outvote it based on offset.

The graph below shows same situation above but analyzed by the rate of change (frequency). It's quite clear from this graph there was a frequency disagreement at 17:23 and that Sourcecheck can invalidate source 0 based on frequency rather than waiting for the offset to grow large enough to cause an issue.



The same issue seen by frequency shows the issue with source 0 (green) much more quickly, just after 17:23:20.

Enabling Sourcecheck

The Sourcecheck feature a global option and cannot be configured per-source. That's done by selecting the 'Sourcecheck' box in the GUI or by setting the following in the configuration file.

```
SOURCE0() { NTPSERVER=...; }
SOURCE1() { NTPSERVER=...; }
SOURCECHECK=1
```

Note that any time sources marked as 'MONITORONLY=1' will not be included in the cross-check of time sources.

Restoring a previously invalid source

Once a time source corrects its time and begins passing Sourcecheck validation it will not be marked 'valid' immediately. Sourcecheck will wait 15 minutes before allowing a source to be considered valid. This avoids switching back to a bad time source because it momentarily shows good time or quickly switching between time sources that are in the 'gray area' of being marked invalid.

Leapsecond checks

In addition to the cluster analysis that Sourcecheck performs it will execute additional checks. To avoid false leap seconds which have plagued other solutions Sourcecheck will confirm that a majority of time sources show a leap second before accepting the leap second as valid. Sourcecheck will

also only permit leap seconds during certain times of the year during which leap seconds are inserted. If a leap second is advertised by a time source outside of these ranges then it is rejected.

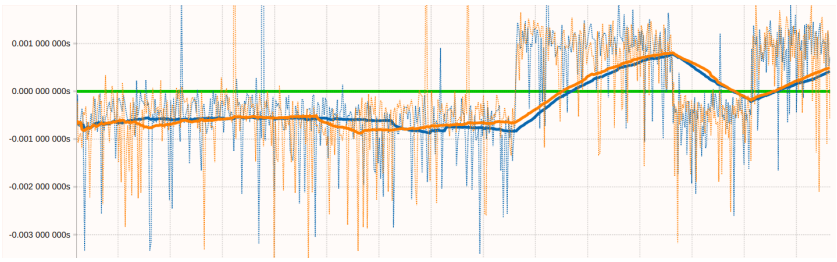
Some configurations can result in unexpected behavior so it's important to be aware of that. If a primary time source is configured to 'slew' in response to a leapsecond but backup time sources are configured to 'step'. In this case immediately after the leap second Sourcecheck would see the primary time source smoothly/slowly slewing the time to correct for the leap second while the backup time sources stepped immediately. That would cause Sourcecheck to reject the primary time source since it disagrees with the two backup time sources. After the primary time source completes the slew it would be accepted by Sourcecheck again as all time sources then agree.

How to setup Sourcecheck properly

The most critical aspect of setting up Sourcecheck is to select good quality time sources. If you select multiple time sources that are outside of your control, are unreliable and cannot be trusted then you have given these sources the ability to invalidate your primary source. So it is always recommended that you use only your own corporate or internal time sources (no public NTP servers). More is always better in this case since Sourcecheck can do some cross-checks with only two sources but three or more are necessary for full validation. In general we recommend 5 sources for a proper configuration.

When all time sources are within a certain threshold (configurable, see 'Tuning' below) then they are all declared to be 'valid' and Sourcecheck will not invalidate any of them. The Sourcecheck threshold setting avoids spurious source changes where all time sources are accurate within a reasonable margin of error (perhaps a few hundred nanoseconds on a decent network). In general it's not necessary to adjust the default value (30 microseconds) but it can be increased to allow inclusion of lower quality sources in the cross-check. For example, a known-high jitter network link or a poor quality time source that has a known quality can be used in with Sourcecheck by adjusting the threshold.

A good practice when configuring Sourcecheck for the first time or with new sources is to first setup the candidate time sources without Sourcecheck enabled. Over a day or two, TimeKeeper will collect data on how the time sources behave. The graph below shows source 0 as a local time source. Sources 1 and 2 are remote time servers. During this test the remote time sources are in agreement but are invalid and noisy because they're both affected by the same shifting network bias. If Sourcecheck had been enabled it would have declared source 0 as invalid since it disagrees with sources 1 and 2 which agree with each other. These cross-check sources would not be good candidates here without additional stable sources that do not use this likely unstable network connection.



Source 0 (green) is correct, but it could be outvoted by sources 1 (blue) and 2 (orange) who noisily agree.

Avoiding loops

Something that must be avoided when configuring Sourcecheck is cycles or loops in a time network. For example, one can configure 3 different systems to cross-check with all other systems. It's possible for a situation to occur where the primary time source is rejected and a backup time source is used on all of these systems. In that case, A might track B, B might track C and C might track A. No system is tracking a valid time but they are all tracking one another which can result in time being incorrect.

Tuning

Due to high network jitter or other variance in network time sources one may want to tune Sourcecheck. For example, it may be necessary to reduce the sensitivity so Sourcecheck won't trigger as easily when a few time sources wander but still catch very large changes. *SOURCECHECK_AUTOVALIDATE_THRESHOLD* is a configuration value that allows this. This value can be described roughly as how tight the sync

between sources must be before they are declared a quorum - or in agreement. The default value is 0.000030 (or 30 microseconds). To change that value to one millisecond:

```
SOURCECHECK_AUTOVALIDATE_THRESHOLD=0.001
```

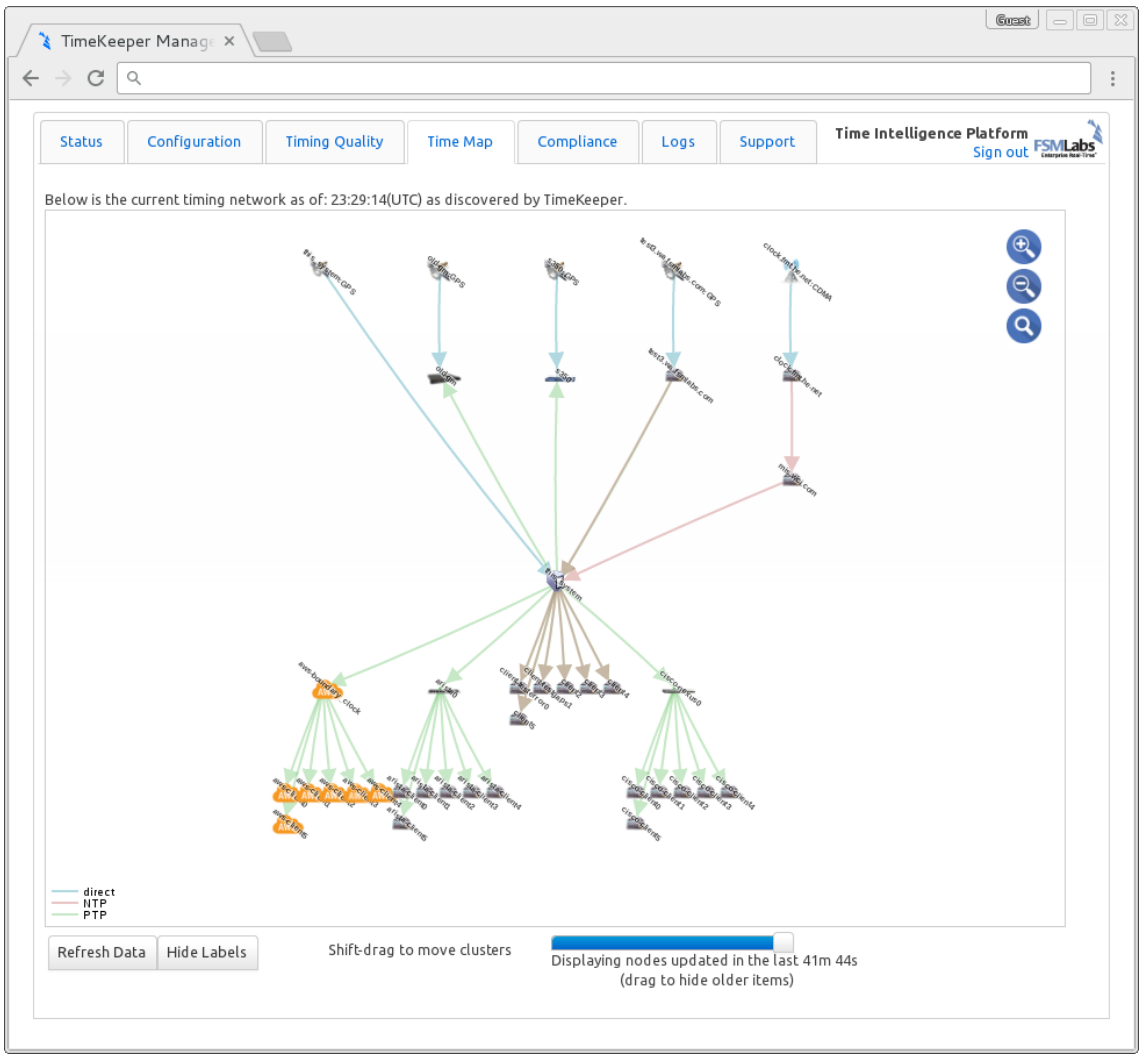
Timing map and network view

TimeKeeper will by default build a map of your timing sources, their timing sources, and so on, as far as it can reach. It'll also include details on any other timing data it sees on the network so you have a single view of what your timing estate looks like in practice.

With this data, you can confirm that you really are getting time from the sources you intend to, with the protocols intended, and that the structure of the network matches your design. Many TimeKeeper users have had the timing map identify issues with their network, like having redundant protocols and paths, only to in the end have a single point of failure upstream. Others use it to provide a quick overview of the current accuracy and connectivity of the systems.

An example

Let's look in closer detail at the map we saw earlier:



This is intended to give you an overview of the network at a glance, but let's look in some detail to be clear. The system running the map is centered in the map as 'this_system'. The connections between it and its sources (going up to stratum 1) are shown by protocol, as is the connectivity to what clients it has downstream.

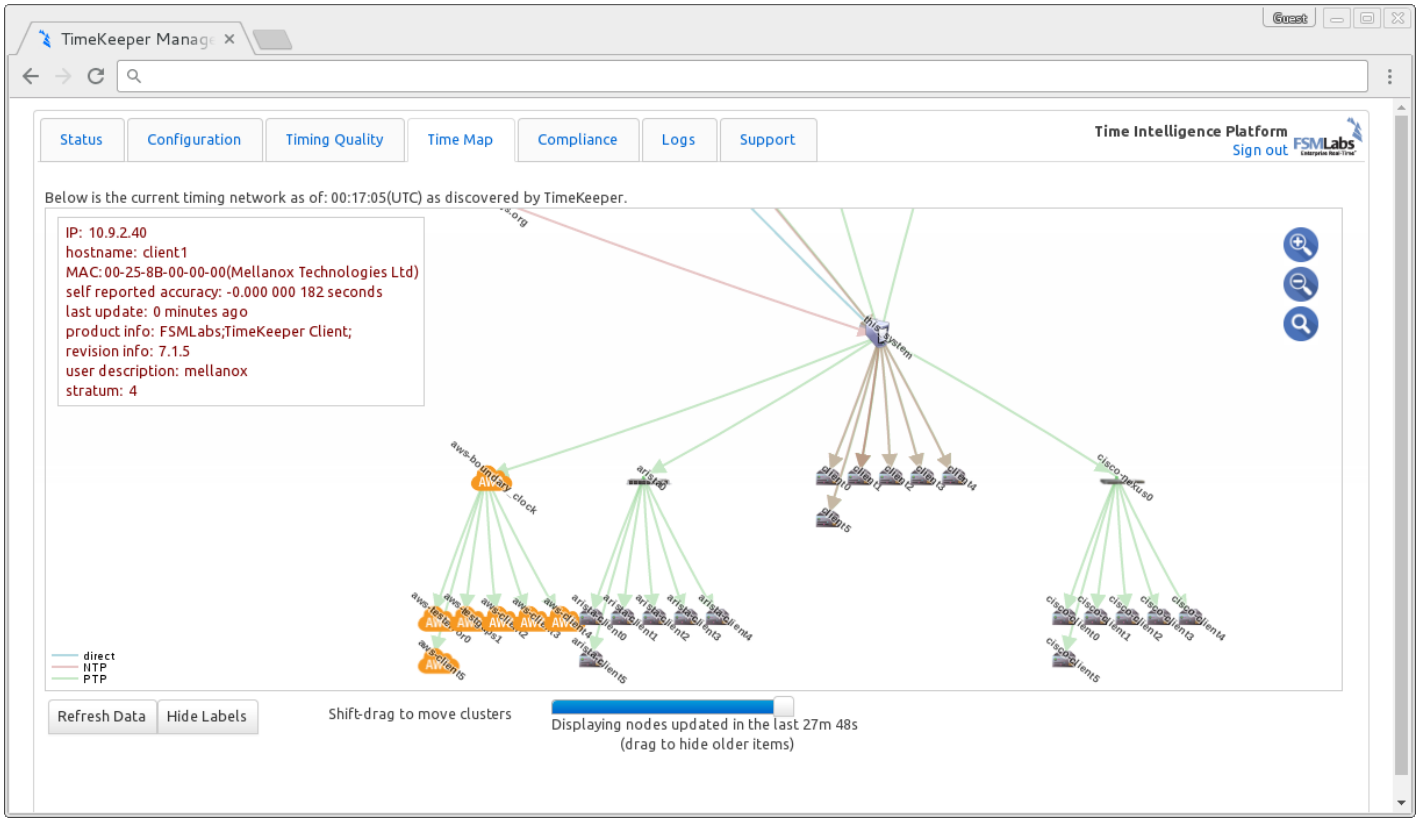
Specifically, we can see:

- This system has a direct connection to a GPS source
- It's getting PTP from a host named 'old_gm' and an S350
- It's also getting NTP from two other hosts
- One of these NTP sources is a stratum 1 server with direct GPS connectivity
- The other NTP source is a stratum server for another host that has a CDMA connection

That's where this system is getting time from. We can also see:

- NTP and PTP are being served downstream
- Some of the clients are hosted in Amazon's EC2 cloud
- Some clients are connected via an Arista switch, others via a Cisco Nexus
- Some clients are directly connected and getting NTP

Interactively, each host can be hovered over for more detail on that particular system. Each arc can be hovered over to get delay details. Clusters can be moved, disabled by age, and more using the controls. Here's an example of a hover over a host:



If there are other timing packets available on the network, but not directly related to this host's sources or clients, it will still be collected and displayed (like multicast PTP data). TimeKeeper Grandmasters collect information from all clients and Grandmasters on the network, so their maps display all of these map details in one place, for a more complete view.

The timing map can show many useful things, but it's more interesting with your own data. It's enabled by default, or by specifying *ENABLE_MAP=1* in */etc/timekeeper.conf* - so you can run it on your network.

Hardware timestamps

Network cards with hardware timestamp capability have clocks on them that timestamp a packet when it arrives. This can greatly improve the accuracy of timestamps because there are no software delays in those timestamps. For those timestamps to be useful the clock on the network card must have good time. TimeKeeper manages the clocks on these devices and steer them to the correct time without any user interaction or configuration.

TimeKeeper transparently detects and take advantages of any network card hardware timestamp features when the operating system, drivers and hardware supports them. Hardware timestamps have been supported with TimeKeeper on Linux for many years, and is now supported on Windows too, starting with Windows 11 and Windows Server 2022. TimeKeeper does not support hardware timestamp features on Solaris.

Enabling hardware timestamps

When running on a platform that supports hardware timestamps there is no additional work needed to configure TimeKeeper to use them. It will enable and verify these features by default if they're enabled in the operating system. You can verify that once TimeKeeper is running with the *tkstatus* command. The output should look something like the below when hardware timestamps are working. The *TIMESTAMPING_HW* indicates that hardware timestamps are used. Column 9 in the data files will show the same value.

If the network card doesn't provide hardware timestamps (or hardware timestamp data isn't usable for some reason), the driver/stack can provide a software timestamp which is indicated by 'TIMESTAMPING_SW'. If the driver doesn't provide either, TimeKeeper falls back to its own internal timestamps, indicated by 'INTERNAL'. Internal timestamps are similar to software timestamps but can be a little noisier because they're taken further up the stack and are subject to more scheduling noise.

```
TimeKeeper Status
================================================================================
TimeKeeper version X.Y.Z
Primary source is: 0
Primary source offset: 0.000000072 seconds
Primary source type: PTP(10.0.0.12:0,unicast,telcom)
Last time update: 0 seconds ago

Src|offset      |source type                    |update age(s)|timestamp type
================================================================================
  0  0.000000072  PTP(10.0.0.12:0,unicast,telcom)            0 TIMESTAMPING_HW
```

The file timekeeper_stamps.log in the configured LOGDIR directory will show a per-source and per-server summary of hardware timestamps. This will show a snapshot of the last 30 seconds of data as well as historical data since the last time TimeKeeper was restarted.

Enabling hardware and software timestamps (Windows-specific)

On Windows the same steps apply as above regarding log files and formats, but hardware and software timestamping may need to be enabled. The below steps in PowerShell (as Administrator) will enable software timestamping for a device named Eth4. For completeness this includes steps required to install components needed for HW/SW timestamping. This is provided as an example only, please refer to the Microsoft documentation for details.

```
PS C:\> Set-ExecutionPolicy remotesigned # Enable execution of below steps in some environments
PS C:\> Install-Module SoftwareTimeStamping
PS C:\> Import-Module SoftwareTimestamping
PS C:\> Get-DscResource -Module SoftwareTimeStamping
PS C:\> Enable-SWTimestamping -NetAdapterName Eth4
PS C:\> Restart-NetAdapter -Name Eth4
```

You can disable software timestamping on a device with:

```
PS C:\> Disable-SwTimestamping -NetAdapterName Eth4
```

Similarly, hardware timestamping may need to be enabled explicitly with the following commands to display, enable, or disable the feature:

```
PS C:\> Get-NetAdapterAdvancedProperty -Name Eth4 -Displayname 'PTP Hardware Timestamp'
PS C:\> Set-NetAdapterAdvancedProperty -Name Eth4 -Displayname 'PTP Hardware Timestamp' -DisplayValue 'Enabled'
PS C:\> Set-NetAdapterAdvancedProperty -Name Eth4 -Displayname 'PTP Hardware Timestamp' -DisplayValue 'Disabled'
```

On Windows either hardware or software timestamping must be enabled on an interface, but not both. So if hardware timestamping is intended make sure to disable software timestamping on the interface, and vice versa if software timestamping is preferred. Having both enabled will not prevent TimeKeeper from using any configured time sources, but will cause INTERNAL timestamps to be used rather than software or hardware.

Network card clock logging

Every update to the network clocks is logged just like updates to the system clock. Instead of the log files being timekeeper_$SOURCE.data where $SOURCE is 0 through 49, 100 through 149 are used. This allows for up to 50 network card clocks to be controlled. The log file format is the same as for any other time source and is described in the "File formats" section.

The accuracy listed is when comparing the network card clock against the system clock. The one-way trip times are when reading the network clock (typically across a PCIe bus). The exception to this is noted below on Linux with the steering nic optimization.

On Windows, the network clocks are managed differently and currently not logged in the .data files as they are on Linux.

Steering NIC optimization (Linux)

Normally TimeKeeper receives a time update and uses that to adjust the system clock. It then will adjust all the network card clocks by steering them to match the system clock. In some cases TimeKeeper may use an optimization.

The optimization is to steer a given network card clock directly to the incoming NTP or PTP feed rather than to the system clock. This can improve the accuracy somewhat since it avoids one extra step (through the system clock). This will only be done when:

- The system clock is very stable and shows low jitter
- The primary NTP/PTP source is stable and shows low jitter
- The primary NTP/PTP source is received in the card being optimized

When the optimization is being used you will see a different message in the data file associated with that network card clock. An additional value will appear in column 12. A 'sourceN' value will appear in which indicates this clock is being directly steered to the Nth time source. If that was source0 then the information may show up as:

```
1635174614.904756863 -0.000000173 -0.000000021 -0.000000152 0.000051843 0.000000000 0.000000000 -0.000000173 N/A 0.000001102 0.000019956 PHC(002590d096d2,freqcorr:-0.000051848,sou
```

When a network clock is in this mode the logs will not show the accuracy, round-trip time and similar in reference to comparing the clock against the system clock but rather the values when comparing the clock against the remote time source. So accuracy is against the remote source and the one-way trip times are over the network to the remote time source.

This optimization can be disabled with the following option in the *timekeeper.conf* file:

```
ENABLE_STEER_NICTOSOURCE=0
```

Hardware timestamp problems

TimeKeeper does significant cross-checks and validation of hardware timestamps as there can be device and/or driver issues that can otherwise cause significant accuracy issues. These errors will be logged in the main timekeeper log file /var/log/timekeeper on Linux and C:\Program Files\timekeeper\var\log\timekeeper.log on Windows. The first indication that there is a hardware timestamp problem is that software or internal timestamps are being used when hardware is expected.

One common error is that a network card/driver provides hardware timestamps for transmit but not receive, or vice-versa. The main TimeKeeper log file will show something similar to:

```
1512507850.617854123: Source 0: timestamp mismatch occurred: sync RX: TIMESTAMPING_HW, delay request TX: TIMESTAMPING_SW, able to correct mismatch
1512507859.276330604: Source 0: timestamp mismatch occurred: sync RX: TIMESTAMPING_HW, delay request TX: TIMESTAMPING_SW, able to correct mismatch
1512507859.277407383: Source 0: timestamp mismatch occurred: sync RX: TIMESTAMPING_SW, delay request TX: TIMESTAMPING_HW, able to correct mismatch
1512507864.693405652: Source 0: timestamp mismatch occurred: sync RX: TIMESTAMPING_HW, delay request TX: TIMESTAMPING_SW, able to correct mismatch
```

The above error is due to a driver/card problem but is being corrected by TimeKeeper. TimeKeeper will fall back to software timestamps if available to avoid asymmetry errors introduced by having hardware timestamps one direction but software the other direction.

Disable hardware timestamps

For testing and avoiding network card/driver bugs it is sometimes necessary to disable hardware timestamps. One can disable hardware timestamps on all network cards with ENABLE_HWTSTAMPS=0 in the TimeKeeper configuration file. This will configure each network card seen to disable (in hardware) the hardware timestamps. This is useful when some older drivers (with bugs) cause kernel issues when hardware timestamps are enabled.

It's also possible to leave the hardware timestamps on in the hardware but on a source-by-source basis simply ignore those hardware timestamps. This is helpful for testing sometimes. This can be done with ENABLE_HWTSTAMPS=0 in each source block to ignore hardware timestamps for that source.

There are yet other cases where it's necessary to disable hardware timestamps on the card itself for specific cards but not others. The above configuration variables won't allow that but AVOID_IFACES=… will. For example, if you have one of the network cards that advertises hardware timestamp support but contains a bug that causes serious networking problems or reboots you may want to use this option. In those cases you can enter those network cards in the AVOID_IFACES= value (comma separated list) and TimeKeeper will entirely ignore these devices. It will not attempt to configure them, will not track the time on them and will not try to steer the time on them. This is one possible work-around for buggy drivers.

TimeKeeper also maintains an internal set of known unstable drivers and avoids versions known to be unstable by default. This is automatically done but can be overridden with the ENABLE_NIC_BLACKLIST configuration option. More details on this and AVOID_IFACES is available in the configuration section, "Global options."

Network asymmetry detection

Network asymmetries arise when packets going from point A to B take more or less time to deliver than they take to get from B to A. This can bias timing data silently - many tools are unable to detect or correct for this kind of behavior, and instead report a great sync when in reality they're many microseconds or more off.

This commonly happens happens in longer network segments such as leased lines or WAN connections. It can also happen when there's a speed mismatch in the network path between two points, like when a 100mbit or 1G time source delivers time to a client via a 10G network link. Switches transmitting data between the nodes can handle the speed mismatch by using store and forward techniques in one direction, but cut through the other way.

Similar behaviors can happen in other network designs also, and can silently shift the clock by many microseconds. TimeKeeper is built to detect this issue so that you can correct or account for it in your network.

Configuring to detect asymmetries

When properly configured TimeKeeper is able to detect and measure network route asymmetry. To do this one needs both systems partcipating in the measurement to be using a time source that you consider very accurate. That is because all measurements will be against that time source and any inaccuracy in it will result in errors in the asymmetry measurement. We normally recommend this be done on either a grandmaster with GPS time or a client of a grandmaster (with GPS) that is getting time directly from GPS or a PPS feed or other accurate time source.

This mode cannot be used to measure asymmetry between a client and its server. That is because the client is tracking the server and they do not have an independent source of time to compare against.

This high quality reference time should be *SOURCE0*. At the other end of the network route that you wish to measure you must have another GPS (or similar) PTP or NTP server. To measure this network route for asymmetry you will configure:

```
SOURCE1() { NTPSERVER=remote_hostname; ENABLE_DETECT_ASYMMETRY=1; }
```

Once TimeKeeper has established a good sync for *SOURCE0* you will see messages in *timekeeper_SOURCE1.asymmetry* in */var/log/* showing the average one way time from the remote source to the local system and from the local system to the remote system.

In a network without an asymmetry, the measured delay in one direction will be the same as the other. If there's an asymmetry though, the degree of it will be shown in these files. From there you can investigate the network link between the sources to identify the source, or use TimeKeeper's configuration to mitigate the issue. Contact support@fsmlabs.com for details.

Major/minor sources

What if you had a very accurate pulse per second (PPS) source you wanted to use for time, but it was only a PPS? There's great information there telling you the boundaries of a second, but nothing indicating the time of day (or month or year).

This happens pretty frequently in testing and validation, where a high quality PPS is delivered, but that host also needs to know the time of day in order to make use of it. That host may be able to get time of day information from a local NTP server. Even if that NTP feed is relatively inaccurate (can't guarantee time to the microsecond/nanosecond level), it can still be used to provide *major time* (time of day to within a second) to match with the *minor time* (accurate information about the nanosecond/microsecond range) provided by the PPS.

Configuring TimeKeeper for major/minor sources

TimeKeeper can be configured to combine these sources into one highly accurate local clock as well as act as a time server on the network. Here is an example of how to configure for this scenario:

```
SOURCE0() {
 PPSDEV=/dev/ttyS1
 MAJORTIME=SOURCE1
}
SOURCE1() {
 NTPSERVER=hostname
}
```

PPSDEV in source 0 here is a local PPS via a serial input, it could instead be a local bus card, as detailed elsewhere in these documents. In this configuration, the primary time source (SOURCE0) is a serial device providing an accurate pulse per second but no time of day information. SOURCE1 is an NTP server with time data accurate to the second. SOURCE0 combines the PPS with the NTP data to provide an accurate primary source. In this configuration, if the PPS is lost, TimeKeeper will fall back to use just the NTP server. (Other sources could be listed in between SOURCE0 and SOURCE1 or after SOURCE1 also.)

Distributing local time

Most of the time, TimeKeeper receives and/or distributes from an authoritative upstream source. That source may be NTP from a server or a direct connection to GPS or PTP from a grandmaster or a PPS, or a number of other sources. No matter what the source is, TimeKeeper tracks it and may serve time based on it.

In some cases though, there may be no external connectivity at all, and all that's present is the local system itself for time. TimeKeeper can take the local clock and use it as the 'source' of time, and use that as a source to steer any time served to clients.

This is common with clusters of systems that need to be tightly synchronized to each other, but that synchronization may not need to match any outside time reference like GPS.

Configuring TimeKeeper to distribute local time

TimeKeeper calls this type of source *self* and it's configured with the PPSDEV parameter. With a primary source declared as *PPSDEV=self*, TimeKeeper will serve the local system time in any configured PTP servers, and will also respond with that time when queried via NTP.

The configuration for this type of source looks like the following, with the ability to serve that time via NTP to any clients:

```
SOURCE0() { PPSDEV=self; }
SERVENTP=1
```

Any clients that query a system with this configuration will get that system's time in the NTP response, and steer their clock to match. This approach does have limitations, but it can keep a set of independent machines very tightly synchronized.

PTP layer 2

To enable PTP layer 2 support in TimeKeeper use the PTP_LAYER2 configuration parameter. When enabled the server or source communicates via 802.3 (raw Ethernet frames/Layer 2 packets).

The PTP_LAYER2 option is available to use as a server or a source parameter. Here is an example of how to configure a server:

```
SERVEPTP0 () { PTPSERVERVERSION='2'; PTPSERVERDOMAIN='0'; PTP_LAYER2=1; }
```

Here is an example of how to configure a source (that can communicate with the server above):

```
SOURCE0 () { PTPDOMAIN='0'; PTPCLIENTVERSION='2'; PTP_LAYER2=1; }
```

Please note the following limitations:

- PTP_LAYER2 is only supported on Linux.
- The PTP_LAYER2 option cannot be used with the IPV6 configuration parameter.
- In order to record layer 2 PTP packets with the VERBOSE_TCPDUMP option, the interface for capture on that server or source needs to be specified using the IFACE parameter. Only one interface can be captured with VERBOSE_TCPDUMP at a time.

PTP Profiles

TimeKeeper allows you to specify PTP profiles with the PTP_PROFILE configuration parameter. The only profile available with the PTP_PROFILE option is *automotive*.

The PTP_PROFILE option is available as a server or a source parameter.

Automotive

The *automotive* profile is based on the IEEE P802.1AS-Rev/D8.0 specification. Here is an example of how to configure a server:

```
SERVEPTP0 () { PTPSERVERVERSION='2'; PTP_PROFILE='automotive'; }
```

Here is an example of how to configure a source (that can communicate with the server above):

```
SOURCE0 () { PTPCLIENTVERSION='2';  PTP_PROFILE='automotive'; }
```

Please note the following limitations for the *automotive* profile:

- The automotive profile is only supported on Linux.
- To record layer 2 PTP packets with the VERBOSE_TCPDUMP option for the *automotive* profile, specify PTP_LAYER2=1; in the configuration. See the "PTP layer 2" section for more details.

Telecom

TimeKeeper supports the telecom profile (full unicast) but not with the PTP_PROFILE option. To specify the telecom profile, provide the address of the Grandmaster using the PTPSERVER option in the source definition.

Here is an example of how to configure a source to use the telecom profile:

```
SOURCE0 () { PTPCLIENTVERSION='2';  PTPSERVER='gm_hostname'; }
```

See the "IEEE 1588 (PTP) tips" section for more details.

Using TimeKeeper

TimeKeeper does a lot of things - operating as a timing client, a timing server, a grandmaster, and so on, supporting all of the standard timing protocols.

To be usable from an embedded scale to enterprise though, it has to do more than implement some protocols. It has to support a wide variety of tools and integrate with many services. Here's a short example list:

- Authentication - RADIUS/TACACS+, ssh, etc.
- Web interface tools for visualization and reporting
- SNMP, syslog, email alerting (including SNMP MIB support)
- Service management

Also we'll go over what add-on tools are provided with TimeKeeper and how to make use of them, TimeKeeper file formats, configuration details, and more.

First let's go over the full set of configuration details available for use in TimeKeeper.

Configuration

This section details all of the options supported by TimeKeeper. These options are all stored in the following file:

Linux:

> /etc/timekeeper.conf

Windows:

> %ProgramFiles%\timekeeper\timekeeper.conf

TimeKeeper's configuration file is made of simple shell statements and functions. On Unix based systems the configuration file must be readable by the "nobody" user.
Time sources are identified with the syntax:

```
SOURCE0() {
 # Source configuration parameters
}
```

The number following the SOURCE function name, 0 here, indicates the priority of the source.

PTP servers are identified similarly:

```
SERVEPTP0() {
 # PTP server configuration parameters
}
```

PTP servers are numbered to differentiate between different servers on different domains, interfaces, etc., but are not priority based like the sources are above.

Compliance report definitions also follow this same form:

```
COMPLIANCEREPORT0() {
 # Compliance report configuration parameters
}
```

Following the shell conventions, capitalization is important, so all functions and variables must be capitalized in order for TimeKeeper to detect them. Any detected errors in the configuration file will be logged in the timekeeper log in ${LOGDIR} during startup.

Individual configuration options relevant to each scenario are discussed below. In the Values column, "N, M" means the option can be set to either N or M, "N - M" means the option can be set to values from N to M, inclusive, and "N -" means the same as "N - M" but there is no maximum value.

Global options

Below are global settings for TimeKeeper. For settings that take an input of 0 or 1, 0 turns the feature off and 1 turns the feature on. In general, TimeKeeper will apply default values, and there is no need to explicitly configure these settings unless there is a need for a specific feature.

Option	Values	Units	Default	Example	Description
ALLOW_SET_TIME_AFTER_STARTUP	0, 1	NA	0	1	Allow TimeKeeper to set the time (perform a step) if the time offset is found to be large after becoming active. When a step is needed, TimeKeeper emits a syslog/SNMP trap event indicating the event occurred.

Option	Values	Units	Default	Example	Description
AVOID_IFACES	text	NA	NA	‘eth4,eth5’	A comma separated list of interface names that TimeKeeper will avoid enabling features on, like timestamping capabilities.
CPU	-1 -	NA	(see Description)	0	The CPU number that TimeKeeper should put all processes and threads on. A value of -1 will prevent TimeKeeper from applying affinities. If left unspecified, TimeKeeper will choose the last CPU on Linux.
DONOTROTATELOGSONSTARTUP	0, 1	NA	0	1	Prevent TimeKeeper from rotating logs on startup.
DISABLE_CLOCK_STEER	0, 1	NA	0	1	A value of 1 prevents TimeKeeper from steering clocks, including the system clock. It will also prevent TimeKeeper from disabling other timing daemons at startup. This allows TimeKeeper to monitor another time daemon. Setting this value to 1 will prevent TimeKeeper from steering the clock and can result in accuracy issues.
EMAILNOTIFICATION	text	NA	NA	‘jsmith@example.com’	A comma separated list of email addresses to send alerts to.
EMAILNOTIFICATION_THROTTLE	0 - 7200	seconds	0	300	Used with EMAILNOTIFICATION. If set, when an event occurs an initial email will be sent. Other emails that would be sent between then and the throttle timeout value will be queued. Once the throttle timeout expires, any queued messages will be delivered as a single bundled email.
ENABLE_COMPLIANCE	0, 1	NA	0	1	A value of 1 will enable the Compliance package.
ENABLE_COMPLIANCE_DNS	0, 1	NA	1	1	A value of 1 will enable the Compliance module to resolve IP addresses to hostnames.
ENABLE_COMPLIANCE _GLOBAL_REPORT	0, 1	NA	1	1	A value of 1 will enable the Compliance report that covers all clients found.
ENABLE_FILESYNC_QUERY	0, 1	NA	0	1	A value of 1 enables queries for the newer Filesync log transfer method if the global ENABLE_MANAGEMENT_QUERY is on as well.
ENABLE_FILESYNC_RESPONSE	0, 1	NA	0	1	A value of 1 enables responses to the newer Filesync log transfer method responses if the global ENABLE_MANAGEMENT_RESPONSE is on as well.
ENABLE_HWTSTAMPS	0, 1	NA	1	1	TimeKeeper enables hardware timestamping by default, and in rare cases this triggers issues with buggy drivers and firmware. Setting this value to 0 will prevent TimeKeeper from trying to enable hardware timestamping at all.
ENABLE_INACTIVE_SOURCE_ALERT	0, 1	NA	0	1	A value of 1 enables TimeKeeper to send alerts when a source is inactive and does not respond within the outage time limit. Alerts for an inactive source are sent once every 3 minutes.
ENABLE_MANAGEMENT_QUERY	0, 1	NA	(see Description)	1	A value of 1 enables the system to query PTP systems for their time sync quality data, which is then stored on the querying system. Individual PTP sources and servers may enable or disable queries independently if this global option is enabled. If disabled it disables queries on all PTP sources and servers. The option defaults to 1 if allowed by the license and TimeKeeper is configured to serve NTP or PTP.
ENABLE_MANAGEMENT_RESPONSE	0, 1	NA	1	1	A value of 1 allows the system to respond to incoming PTP management data requests. Individual PTP sources and servers may enable or disable the ability to respond independently if this global option is enabled. If disabled it disables responses on all PTP sources and servers. The option defaults to 1 if allowed by the license and PTP is configured.
ENABLE_MAP	0, 1	NA	1	1	A value of 1 enables the collection of information needed to build a map of the timing network.
ENABLE_NIC_BLACKLIST	0, 1	NA	1	1	A value of 1 allows TimeKeeper to avoid enabling features on interfaces with drivers that are known to be unstable. When applied TimeKeeper will use the interface as it would normally, but without all hardware specific features enabled. Disabling ENABLE_NIC_BLACKLIST will allow TimeKeeper to enable features even if the driver and version are known to be unstable.
ENABLE_NTP_FOLLOWUP	0, 1	NA	1	1	A value of 1 will cause the TimeKeeper NTP server (if enabled) to also respond to NTP requests with a followup NTP packet. On TimeKeeper clients, this allows processing of the followup to improve timing accuracy.

Option	Values	Units	Default	Example	Description
ENABLE_PREFILESYNC _MANAGEMENT_RESPONSE	0, 1	NA	1	1	A value of 1 enables responses to queries from the management data transfer method that preceded Filesync. This should be disabled when all client data transfer is intended to be done with Filesync.
ENABLE_SATELLITEDATA	0, 1	NA	1	1	A value of 1 enables the collection of detailed GPS satellite angle/strength data if the GPS device supports it.
ENABLE_SFC_UUID_FILTER	0, 1	NA	1	1	When tracking PTP with a hardware timestamping Solarflare card, a value of 1 will configure the card to only provide hardware timestamps for PTP traffic that matches the UUID of the highest priority PTP source.
ENABLE_SOURCESYSLOG	0, 1	NA	0	1	A value of 1 enables the collection of time source data into syslog.
ENABLE_TKGPS_DETAIL	0, 1	NA	1	1	A value of 1 enables the collection of detailed GPS state with a TimeKeeper GPS device, if present - such as on a TimeKeeper grandmaster.
ENABLE_WEB_MANAGEMENT	0, 1	NA	0	1	A value of 1 enables the web management tools on the server.
FILESYNC_END_TIME	00:00:00 - 24:00:00	NA	24:00:00	09:00:00	A text field formatted as 'HH:MM:SS' (hours, minutes, seconds) offset from UTC midnight indicating when the Filesync method of transferring client log files between servers and clients is required to end globally. This value can be overridden in specific sources' or servers' configurations, the default value '00:00:00' is equivalent to '24:00:00', that is, the end of the day.
FILESYNC_EXCLUDE	text	NA	NA	'192.168.*'	A text field determining which Filesync clients/directories are NOT shared by the system using standard file glob patterns (*, ?, [], and !). This field defaults to allowing all clients/directories to be synchronized with a remote system. See the advanced networking section of the TimeKeeper manual for more information.
FILESYNC_INCLUDE	text	NA	*	'???.?[1-7].12.*'	A text field determining which Filesync clients/directories are shared by a system using standard file glob patterns (*, ?, [], and !). Inclusion takes priority over exclusion. This field defaults to allowing all clients/directories to be synchronized with a remote system. See the advanced networking section of the TimeKeeper manual for more information.
FILESYNC_START_TIME	00:00:00 - 24:00:00	NA	00:00:00	08:00:00	A text field formatted as 'HH:MM:SS' (hours, minutes, seconds) offset from UTC midnight indicating when the Filesync method of transferring client log files between servers and clients is allowed to start globally. This value can be overridden in specific sources' or servers' configurations. Together with 'FILESYNC_END_TIME' it defines a range during which transfers may occur.
HTTPS_KEY_PASSPHRASE	text	NA	NA	'HlrkiGkXEvI7Hkl 0wGOuzw=='	A string containing the encrypted password to be used to access a password-protected private key when enabling HTTPS support. The passphrase should be encoded with the encodepassphrase tool provided with TimeKeeper.
INACTIVE_CLIENT_THRESHOLD	0 -	seconds	0	100	Specifies how long a client should be inactive before an alarm is triggered. (e.g., INACTIVE_CLIENT_THRESHOLD=600 sends alarms if any client inactivity exceeds 10 minutes). A value of 0 disables the alarm.
INITIAL_SERVE_ACCURACY	0.0 -	seconds	0.0	0.00001	Accuracy required before serving time via NTP/PTP/TIME protocols after startup.
IPTOS	0 - 255	NA	16	56	The default server IP ToS value as an integer. For conversion to DSCP, use standard conversions. This has no effect on Windows.
LOGDIR	text	NA	(see Description)	'/var/log/timekeeper/'	A string naming the directory where TimeKeeper logs will be stored. On non-Grandmaster Linux hosts the default is /var/log. On Windows the default is %ProgramFiles%\timekeeper\var\log. The string must be a full path without spaces. To use the Windows default location, do not configure this parameter.
MANAGEMENT_QUERY_INTERVAL	15.0 - 3600.0	seconds	60.0	15.0	Indicates how frequently to query PTP clients for their time sync quality data.

Option	Values	Units	Default	Example	Description
NTPSYNCERRORTHRESHOLD	0.0 -	seconds	0.0	0.000010	If management queries and SNMP traps or email alerts are enabled, any NTP clients reporting a sync exceeding this threshold will cause an alarm to be triggered. (e.g., NTPSYNCERRORTHRESHOLD=0.000010 sends alarms if any client sync quality exceeds +-10us). A threshold violation will only be reported after the error has been seen to be below the threshold first. A special value of 0 disables the alarm and is the default.
SERVENTP	0, 1	NA	0	1	A value of 1 will cause TimeKeeper to respond to NTP requests on all interfaces.
SERVENTP_ENABLE _MANAGEMENT_QUERY	0, 1	NA	1	1	A value of 1 allows collection of self-reported accuracy information from clients via NTP. TimeKeeper must also be configured to send management queries. Note that if the client is TimeKeeper and can communicate via PTP, the NTP data will be superseded with the more complete details via PTP.
SERVENTP_IFACE	text	NA	NA	'eth4'	A string naming the network interface that TimeKeeper should respond to NTP queries on. If TimeKeeper receives a request via that named interface, it will respond, but only on that interface. If the value of this setting is empty, TimeKeeper listens and responds on all interfaces.
SERVENTP_NTPKEYIDS	1 - 65535	NA	NA	1,3,10	A comma separated list of MD5 key IDs read from */etc/ntp/keys* (on Windows, *%ProgramData%\timekeeper\ntp.keys*) used to authenticate client requests.
SERVETIME	0, 1	NA	0	1	A value of 1 will cause TimeKeeper to respond to time requests (RFC 868) on all interfaces.
SET_TIME_ON_STARTUP	0, 1	NA	0	1	A value of 1 forcibly sets the time on startup regardless of offset, rather than trying to slew, for a faster sync on startup. A value of 0 will allow the TimeKeeper to slew in as long as the offset is less than +/-5 seconds from the primary time source unless SET_TIME_THRESHOLD is used to set a different value.
SET_TIME_THRESHOLD	0.01 -	seconds	5.0	0.5	Specifies the initial offset threshold allowed before a step is taken on startup, and also the threshold to allow an immediate correction post-startup if ALLOW_SET_TIME_AFTER_STARTUP is set. Default value is 5 seconds. This is useful in virtual environments where VM processing delays can cause the clock to have a large offset and need a near immediate rather than slow correction once the offset is confirmed. Minimum value is 10 milliseconds (0.01). Some environments may require a higher minimum value in practice.
SNMPTRAPEID	text	NA	(see Description)	"1234"	The SNMPv3-specific engine ID. For example, a value of "1234" results in "8000A6ED0431323334". If not set, TimeKeeper will base it on the MAC address of the default network interface.
SNMPTRAPHOST	text	NA	NA	10.10.0.12,10.0.0.12	A comma separated list of SNMP servers to send traps to.
SNMPTRAPOID	text	NA	1.3.6.1.4.1.42733.0	1.2.3.4	An OID string to use for sending all SNMP trap messages. If left unspecified, TimeKeeper will deliver traps as specified in the TimeKeeper MIB. This is a legacy configuration option.
SNMPTRAPPASSPHRASE	text	NA	NA	'b5xNNXVAnOalxvq Tz8o35A=='	An encrypted passphrase to be used with SNMPv3 traps.
SNMPTRAPUSERNAME	text	NA	NA	timekeeper	A username to be used with SNMPv3 traps.
SOURCECHECK	0, 1	NA	0	1	A value of 1 will enable timing source cross checks described elsewhere in this document.
SOURCECHECK_AUTOVALIDATE _THRESHOLD	0.0 -	seconds	0.00003	0.001	Sources must agree to within this value for the Sourcecheck feature to allow source validation.
SPECTRACOM_DRIVER_PATH	text	NA	NA	'/tmp/spectracomtsync/'	A string naming the directory where the driver for a TSync card resides, if TimeKeeper intends to use a TSync card at runtime.
STEP_ON_LEAPSECOND	0, 1	NA	0	1	A value of 1 corrects leap seconds with an immediate clock jump instead of a slew.

Option	Values	Units	Default	Example	Description
SYMMETRICOM_DRIVER_PATH	text	NA	NA	'/opt/bc637_driv_rel'	A string naming the directory where the driver for a BC637 card resides if TimeKeeper intends to use a BC637 card at runtime. This is the path to the directory that contains the drvr/ and samples/ directories, not the drvr/ directory itself.
SYNC_ERROR_THRESHOLD _ALERT_ONLY_PRIMARY	0, 1	NA	0	1	A value of 1 enables TimeKeeper to send sync error threshold alerts only for the primary source. By default alerts are sent for all sources that exceed the sync error threshold.
SYNC_ERROR_THRESHOLD _THROTTLE	0 - 3600	seconds	5	3600	Throttles the rate that TimeKeeper will send alerts about sync error threshold values being exceeded. The default prevents TimeKeeper from sending alerts more often than every 5 seconds.
TLS_VERSION	1.2, 1.3	NA	(see Description)	1.2	Specifies TLS version to use. Specify '1.2' to use TLS version 1.2 and limit ciphers in use for the TimeKeeper GUI, or '1.3' to use TLS version 1.3. The default is to not limit TLS versions in use. '1.3' will only appear on devices that support TLS version 1.3.
VERBOSE_BMC	0, 1	NA	0	1	A value of 1 enables verbose logging of any BMC algorithm activity when handling PTP data.
VERBOSE_COMPLIANCE	0, 1	NA	1	1	A value of 1 enables verbose logging of Compliance-related activity.
VERBOSE_MANAGEMENT	0, 1	NA	0	1	A value of 1 enables verbose logging of management message behavior.
VERBOSE_MAP	0, 1	NA	0	1	A value of 1 enables verbose logging of timing map operations.
VERBOSE_NMEA	0, 1	NA	0	1	A value of 1 enables verbose logging of NMEA handling.
VERBOSE_NTP	0, 1	NA	0	1	A value of 1 enables verbose logging of NTP behavior.
VERBOSE_PPS	0, 1	NA	0	1	A value of 1 enables verbose logging of PPS-based configurations.
VERBOSE_PTP	0, 1	NA	0	1	A value of 1 enables verbose logging of PTP behavior.
VERBOSE_SERVER	0, 1	NA	0	1	A value of 1 enables verbose logging of TimeKeeper server actions.
VERBOSE_SOCKET	0, 1	NA	0	1	A value of 1 enables verbose logging of socket-related activity.
VERBOSE_TCPDUMP	0, 1	NA	0	1	A value of 1 enables tcpdump recording of PTP and NTP, kept alongside other TimeKeeper logs.
VERBOSE_TIMESTAMPS	0, 1	NA	0	1	A value of 1 enables verbose logging of timestamping activity.
WEB_MANAGEMENT_IP	0.0.0.0 - 255.255.255.255	NA	0.0.0.0	10.10.10.1	An IP address that will restrict serving web data to clients accessing the machine via that IP. The default is to listen on all IPs.
WEB_MANAGEMENT_PORT	-1 - 65535	NA	-1	8082	What port the web management should run on. HTTP defaults to off (WEB_MANAGEMENT_PORT=-1) and can be enabled by specifying a port value.
WEB_MANAGEMENT_TIMEOUT	5 -	minutes	44640	1440	How long before a web interface session is automatically logged out (idle or not).
ENABLE_PAM_AUTH	0, 1	NA	0	1	A value of 1 enables the web management to authenticate with the PAM user name and password from the Linux host. ENABLE_PAM_AUTH is only supported on Linux.
WINROTATECOUNT	-1 -	days	7	10	Sets the amount of TimeKeeper logs to retain on Windows. A value of -1 prevents TimeKeeper from ever rotating logs.

Per source options

Examples of sources include an NTP server, a PTP grandmaster, or a local bus card. Each option will have different configuration directives.

NTP source options

Option	Values	Units	Default	Example	Description
ENABLE_DETECT_ASYMMETRY	0, 1	NA	0	1	Will detect, measure, and notify the user of network link asymmetries.

Option	Values	Units	Default	Example	Description
ENABLE_RERESOLVE	0, 1	NA	0	1	A value of 1 will cause the source to periodically re-resolve the DNS name specified for the NTP source.
NTPKEYID	1 - 65535	NA	NA	5	Causes TimeKeeper to read */etc/ntp/keys* (on Windows, *%ProgramData%\timekeeper\ntp.keys*) for a matching MD5 Key ID to use for authentication with the NTP server.
NTPSERVER	text	NA	NA	'ab-1.example.net'	The IP address or DNS name of the intended NTP server.
NTPSYNCRATE	0.015625 - 128	pps	0.9	1.0	The rate at which to make NTP requests, X queries every second. NTPSYNCRATE=0.5 causes TimeKeeper to query the NTP server every 2 seconds.

PTP source options

Option	Values	Units	Default	Example	Description
ALLOW_DROPPED_FOLLOWUP	0, 1	NA	0	1	A value of 0 means a PTP grandmaster that promises a followup message must deliver one in order for this source to use a given time update. If set to 1, a failed followup delivery won't prevent the sample from being updated. The default is recommended in nearly all deployments, as missing followups indicate a problem with PTP delivery or the grandmaster.
ALLOW_UNREASONABLE_UTC	0, 1	NA	0	1	Forces TimeKeeper to accept sync messages having UTC offsets not known to be correct.
ENABLE_CORRECTION	0, 1	NA	1	1	A value of 0 means this source will not make use of any transparent clock information provided in the PTP data, otherwise it will apply the correction to the timing data.
ENABLE_DETECT_ASYMMETRY	0, 1	NA	0	1	When configured properly, will detect, measure, and notify the user of network link asymmetries.
ENABLE_FILESYNC_QUERY	0, 1	NA	(see Description)	1	A value of 1 causes TimeKeeper to query clients for management data using the newer Filesync log transfer method on this source. Defaults to global option. TimeKeeper sends Filesync queries if both source and global options are enabled and this source is enabled to send management queries.
ENABLE_FILESYNC_RESPONSE	0, 1	NA	(see Description)	1	A value of 1 enables responses to the newer Filesync log transfer method on this source. Defaults to global option. TimeKeeper responds to Filesync queries if both source and global options are enabled and this source is enabled to respond to management queries.
ENABLE_MANAGEMENT_QUERY	0, 1	NA	(see Description)	1	A value of 1 enables this particular source to query PTP systems for their time sync quality data. Defaults to global option. Please see the ENABLE_MANAGEMENT_QUERY option in the global option section for more details.
ENABLE_MANAGEMENT_RESPONSE	0, 1	NA	(see Description)	1	A value of 1 allows this particular source to respond to incoming PTP management data requests. Defaults to global option. Please see the ENABLE_MANAGEMENT_RESPONSE in the global option section for more details.
ENABLE_PREFILESYNC _MANAGEMENT_RESPONSE	0, 1	NA	(see Description)	1	A value of 1 enables responses to PTP management queries from the management data transfer method that preceded Filesync. Only enable this if your client should respond to both Filesync and the older management query method. Defaults to global option. TimeKeeper responds in this way if both source and global options are enabled and this source is enabled to respond to Filesync management queries.
FILESYNC_END_TIME	00:00:00 - 24:00:00	NA	24:00:00	09:00:00	A text field formatted as 'HH:MM:SS' (hours, minutes, seconds) offset from UTC midnight indicating when the Filesync of transferring client log files between servers and clients is required to end for this source. This overrides the global value and the default is equivalent to '24:00:00', that is, the end of the day.
FILESYNC_START_TIME	00:00:00 - 24:00:00	NA	00:00:00	08:00:00	A text field formatted as 'HH:MM:SS' (hours, minutes, seconds) offset from UTC midnight indicating when the Filesync method of transferring client log files between servers and clients is allowed to start for this source. This overrides the global value. Together with 'FILESYNC_END_TIME' it defines a range during which transfers may occur.
IFACE	text	NA	NA	'eth5.5'	This source should watch the named interface for PTP data, where the interface name corresponds with a Linux network device name.
IPTOS	0 - 255	NA	16	128	The IP ToS value as an integer. For conversion to DSCP, use standard conversions. This has no effect on Windows.
PTP_P2P	0, 1	NA	0	1	If set to 1, the client uses P2P, the peer to peer delay measurement mechanism. Otherwise, the client uses E2E, the end to end delay measurement mechanism.
PTP_PROFILE	text	NA	NA	'automotive'	The PTP profile to use in communicating with the PTP server. Only the automotive profile can be specified.

Option	Values	Units	Default	Example	Description
PTP_SYNC_RATE	0.015625 - 128	pps	1	4	The rate that the client requests the PTP server to send telecom-profile sync messages. Actual rate rounds to nearest power of 2 towards 1. For example, a value of 5 results in 4 sync messages per second and a value of 0.7 results in 1 sync message per second.
PTPCLIENTVERSION	1, 2	NA	NA	2	Most users will want version 2.
PTPDOMAIN	0 - 255	NA	0	10	The domain that TimeKeeper will watch for PTP data on.
PTPSERVER	text	NA	NA	10.0.0.90	The IP or DNS name of the unicast PTP server to be used.
PTP_LAYER2	0, 1	NA	0	1	If set to 1, the client communicates via 802.3 (raw Ethernet frames/Layer 2 packets), otherwise will default to communicating via IPv4. PTP_LAYER2 is only supported on Linux. Cannot be used with the IPV6 source configuration parameter.
TTL	0 - 255	hops	1	22	How long a PTP packet should live when routed.
UNICAST	0, 1	NA	0	1	A value of 1 enables unicast delay requests back to the server. Ignored if PTPSERVER is specified.

Local bus device options

Option	Values	Units	Default	Example	Description
BAUD	4800, 9600, 19200, 38400, 57600, 115200	bps	NA	(see Description)	Override default baud rate for this device. Default is 9600 for Spectratime, 115200 for Jackson Labs and u-blox, and 4800 otherwise.
DISABLE_GPS	0, 1	NA	0	1	Use other sources to steer the oscillator for holdover, disabling the GPS.
GNSS_BEIDOU	0, 1	NA	0	1	If set to 1, use of the BeiDou GNSS is enabled on the TimeKeeper GNSS device or PPSDEV device named. If BeiDou and GLONASS are enabled along with GPS and/or Galileo, BeiDou will be disabled because the GNSS device cannot support that combination. Only applicable if TKGPS on a second generation TimeKeeper grandmaster, TKJS or UBLOX is set to 1.
GNSS_GALILEO	0, 1	NA	0	1	If set to 1, use of the Galileo GNSS is enabled on the TimeKeeper GNSS device or PPSDEV device named. If no GNSS is enabled, Galileo will be enabled by default for TK GNSS GM. Only applicable if TKGPS on a second generation TimeKeeper grandmaster, TKJS or UBLOX is set to 1.
GNSS_GLONASS	0, 1	NA	0	1	If set to 1, use of the GLONASS GNSS is enabled on the TimeKeeper GNSS device or PPSDEV device named. Only applicable if TKGPS on a second generation TimeKeeper grandmaster, TKJS or UBLOX is set to 1.
GNSS_GPS	0, 1	NA	0	1	If set to 1, use of the GPS GNSS is enabled on the TimeKeeper GNSS device or PPSDEV device named. If no GNSS is enabled, GPS will be enabled by default. Only applicable if TKGPS on a second generation TimeKeeper grandmaster, TKJS or UBLOX is set to 1.
HOLDOVER_LIMIT	0.0 -	seconds	7200.0	60	How long to remain in holdover before cross checking time against other sources.
PIN	0 -	NA	0	6	Specify a 'pin' for PPS input/output.
PPSDEV	text	NA	NA	'/dev/ttyS0'	The name of the device to be used. Refer to other sections of this document for details in what names can be used.
PPSOUT	0, 1	NA	0	1	For a PPS input/output device configure specified pin as 'output' rather than 'input'.
PPSSOURCE	text	NA	NA	'tkgpio,eth4,0'	An alternative name/location to get a PPS from.
TKGPS	0, 1	NA	0	1	If set to 1, like on a TimeKeeper grandmaster appliance, the source will be assumed to be a TimeKeeper GNSS/GPS device.
TKJS	0, 1	NA	0	1	If set to 1, the PPSDEV device named will be assumed to be a Jackson Labs GPS/GNSS device. GPS is the default.
TKPPS	0, 1	NA	0	1	Configures the source as a TimeKeeper PPS in/out card.
UBLOX	0, 1	NA	0	1	If set to 1, the PPSDEV device named will be assumed to be a u-blox GNSS device.

General source options

There are some generally applicable configuration options that may apply to any source. Please refer to other sections of this document for specifics on their use:

Option	Values	Units	Default	Example	Description
CABLEDELAY	(any decimal)	seconds	0.0	0.000000271	This can be used to correct timing biases induced by long GPS cable runs or other known causes of signal delay. This value is added to the incoming timing data, so with a long cable run that delays signal by 1 millisecond, use CABLEDELAY=0.001 to make TimeKeeper add 1ms to account for it.

Option	Values	Units	Default	Example	Description
ENABLE_HWTSTAMPS	0, 1	NA	1	1	TimeKeeper enables and uses hardware timestamping by default. Setting this value to 0 for a specific source will prevent TimeKeeper from using hardware timestamps, but only for that source.
IPV6	0, 1	NA	0	1	If set to 1, TimeKeeper communicates via IPv6 for that source, otherwise will default to communicating via IPv4. IPv6 is only supported on Linux.
LABEL	text	NA	NA	'GrandMaster One NY'	User defined string for easy reference.
LOWCPU	0, 1	NA	0	1	If set to 1 low CPU consumption is prioritized over accuracy and better noise rejection. The result is much lower CPU overhead and slight reduction in accuracy.
LOWQUALITY	0, 1	NA	0	1	A value of 1 means TimeKeeper will treat the source as low quality and apply additional filtering as well as using more data from the time source. When set this will also cause TimeKeeper to prioritize small and smooth adjustments over aggressively tracking the time from this source so it will react more slowly to erratic changes in the time source. Note that this will increase CPU consumption. See the slew section for more detail.
MAJORTIME	text	NA	NA	SOURCE3	This source should get major (non-sub-second) timing data from the named source. Provide the full name of the source that will provide time of day information.
MONITORONLY	0, 1	NA	0	1	TimeKeeper will track the time source for informational purposes only, but will not allow the source to set the time on the local host if set to 1.
SYNCERRORTHRESHOLD	0.0 -	seconds	0.0	0.0001	If the source's sync quality exceeds this threshold, send an alarm. (e.g., SYNCERRORTHRESHOLD=0.000010 sends alarms if the sync quality exceeds +- 10us). A threshold violation will only be reported after the error has been seen to be below the threshold first. A special value of 0 disables the alarm and is the default.

Per PTP server options

When creating a PTP server, these options may apply:

Option	Values	Units	Default	Example	Description
ENABLE_FILESYNC_QUERY	0, 1	NA	(see Description)	1	A value of 1 enables queries for the newer Filesync log transfer method for this server. Defaults to global option. TimeKeeper sends Filesync queries if both server and global options are enabled and this server is enabled to send management queries.
ENABLE_FILESYNC_RESPONSE	0, 1	NA	(see Description)	1	A value of 1 enables responses to the newer Filesync log transfer method responses. Defaults to global option. TimeKeeper responds to Filesync queries if both server and global options are enabled and this server is enabled to respond to management queries.
ENABLE_MANAGEMENT_QUERY	0, 1	NA	(see Description)	1	A value of 1 enables this particular server to query PTP systems for their time sync quality data. Defaults to global option. Please see the ENABLE_MANAGEMENT_QUERY option in the global option section for more details.
ENABLE_MANAGEMENT_RESPONSE	0, 1	NA	(see Description)	1	A value of 1 allows this particular server to respond to incoming PTP management data requests. Defaults to global option. Please see the ENABLE_MANAGEMENT_RESPONSE option in the global option section for more details.
ENABLE_PREFILESYNC _MANAGEMENT_RESPONSE	0, 1	NA	(see Description)	1	A value of 1 enables responses to PTP management queries from the management data transfer method that preceded Filesync. Only enable this if your client should respond to both Filesync and the older management query method. Defaults to global option. TimeKeeper responds in this way if both server and global options are enabled and this server is enabled to respond to Filesync management queries.
FILESYNC_END_TIME	00:00:00 - 24:00:00	NA	24:00:00	09:00:00	A text field formatted as 'HH:MM:SS' (hours, minutes, seconds) offset from UTC midnight indicating when the Filesync of transferring client log files between servers and clients is required to end for this server. This overrides the global value and the default is equivalent to '24:00:00', that is, the end of the day.
FILESYNC_START_TIME	00:00:00 - 24:00:00	NA	00:00:00	08:00:00	A text field formatted as 'HH:MM:SS' (hours, minutes, seconds) offset from UTC midnight indicating when the Filesync method of transferring client log files between servers and clients is allowed to start for this server. This overrides the global value. Together with 'FILESYNC_END_TIME' it defines a range during which transfers may occur.

Option	Values	Units	Default	Example	Description
IFACE	text	NA	NA	‘bond0’	This server should use the named interface for PTP data, where the interface name corresponds with a Linux network device name.
IPTOS	0 - 255	NA	16	184	The IP ToS value as an integer. For conversion to DSCP, use standard conversions. This has no effect on Windows.
IPV6	0, 1	NA	0	1	A value of 1 enables this server to distribute time via multicast IPv6 to clients, otherwise it will default to multicast IPv4. NOTE: By default PTP servers are enabled for both IPv4 and IPv6 unicast traffic. IPv6 is only supported on Linux.
LABEL	text	NA	NA	‘Lan Segment 10.x Domain 0’	User defined string for easy reference.
MINCLOCKACCURACY	0.0 -	seconds	0.0	0.010000000	Floor for the accuracy advertised by the PTP server. This can be useful for clients that misbehave and switch grandmasters too quickly.
PTP_OVERRIDE_BC_PRIORITY	0, 1	NA	0	1	Allows a PTP server to over-ride the PTP priority 1 and 2 fields. Normally those values are taken from the upstream PTP source. When this option is set to 1 the priority fields are set from PTPPRIORITY1 and PTPPRIORITY2 rather than the upstream PTP source.
PTPPRIORITY1 and PTPPRIORITY2	0 - 255	NA	128	249	These values are broadcast by the PTP server with announce messages.
PTPSERVERDOMAIN	0 - 255	NA	0	6	This indicates which domain the PTP server will broadcast on.
PTPSERVERSYNCRATE	0.015625 - 128.0	pps	1.0	2.0	The rate to multicast PTP sync and announce messages in packets per second. PTPSERVERSYNCRATE=1 causes TimeKeeper to multicast a sync every second (a value of 2 will send two sync messages every second). NOTE: Prior to TimeKeeper version 8.0.12, PTPSERVERSYNCRATE determined both the multicast and telecom sync and announce rates at the server, and the server ignored any client-requested rates. Starting with 8.0.12, PTPSERVERSYNCRATE still specifies the multicast sync and announce rates, but the server now obeys the client-requested rates. If you had been setting PTPSERVERSYNCRATE on the server to specify the telecom rates for all clients, you will now have to configure your clients so that *they* request the correct, desired rates.
PTPSERVERTTL	0 - 255	hops	1	2	How long a multicast PTP packet should live when routed. (A value of 0 effectively disables transmission of multicast PTP messages from this server.)
PTPSERVERVERSION	1, 2	NA	NA	2	Most customers will want version 2.
PTP_LAYER2	0, 1	NA	0	1	If set to 1, the client communicates via 802.3 (raw Ethernet frames/Layer 2 packets), otherwise will default to communicating via IPv4. PTP_LAYER2 is only supported on Linux. Cannot be used with the IPV6 server configuration parameter.
PTP_PROFILE	text	NA	NA	‘automotive’	The PTP profile the server will use for communication. Only the automotive profile can be specified.
SYNCERRORTHRESHOLD	0.0 -	seconds	0.0	0.0001	If management queries and SNMP traps or email alerts are enabled, any clients reporting a sync exceeding this threshold will cause an alarm to be triggered. (e.g., SYNCERRORTHRESHOLD=0.000010 sends alarms if any client sync quality exceeds +-10us). A threshold violation will only be reported after the error has been seen to be below the threshold first. A special value of 0 disables the alarm.

Option	Values	Units	Default	Example	Description
UNICAST	0, 1	NA	0	1	If set to 1, the server will provide unicast delay responses, whether the client provided a unicast or a multicast delay request. Otherwise, the server will respond with the same type of message as the request (multicast response for a multicast request, unicast response for a unicast request).

Compliance report options

When creating a Compliance report, these options may apply:

Option	Values	Units	Default	Example	Description
ALERTTHRESHOLD	0.0 -	seconds	(see Description)	0.00005	Clients that exceed this value will be flagged as in an alert state and out of compliance. Not specifying this value disables the feature.
CLIENTSET	text	NA	NA	'*-ldn.domain.com'	A string identifying which hosts should be included in this report, specified as one or a series of glob-like patterns separated by commas. Clients will be included in the report if their IP address or hostname match one of the patterns.
DAYS	text	NA	NA	1,2,3,4,5	Specifies the days on which the report is to include data specified as each of the numbers 0-6 inclusive where 0 is Sunday and 6 is Saturday. The default, empty, is to include all days and is equivalent to "0,1,2,3,4,5,6". DAYS can also be specified as "Sun,Mon,Tue,Wed,Thu,Fri,Sat" for their English day short names.
ENDTIME	00:00:00 - 48:00:00	NA	NA	21:00:00	Specifies offset from midnight (00:00:00 hours) UTC that the report will END in its coverage of the day. A 0 value indicates ending at the following midnight UTC. STARTTIME and ENDTIME can cover a period of time up to 24 hours. Please refer to the Compliance configuration documentation for details and examples.
ENDTOENDACCURACY	0, 1	NA	0	1	If set, Compliance will include the reported upstream accuracy in any offset threshold calculations.
EXCLUDEDCLIENTSET	text	NA	NA	'www.*'	Which hosts should not be included in this report, specified as one or a series of glob-like patterns separated by commas. Clients will be excluded from the report if their IP address or hostname match one of the patterns. Exclusion takes precedence over inclusion so if a host is included by "CLIENTSET" but excluded by "EXCLUDEDCLIENTSET" it will not be included in the report.
MINALERTCOUNT	0 -	alerts	0	10	Specifies the minimum number of alerts that a client must have across all sources within the configured day (based on UTC midnight or STARTTIME/ENDTIME) in order for alerts to be included in the audit. A value of 0 means all violations are reported. A client with fewer alerts than this value will not have its alerts recorded/reported on; otherwise each alert will be reported in full. Please refer to the Compliance configuration documentation for details.
MINALERTLENGTH	0 -	seconds	0	1	A client exceeding the alert threshold will only be flagged as being out of compliance if it stays above the threshold for MINALERTLENGTH seconds. A value of 0 means any single sample above the alert threshold will constitute a violation.
MINGAPLENGTH	15 - 86400	seconds	180	15	A client missing data for more than MINGAPLENGTH seconds will be flagged as having a gap in its data.
MINWARNINGLENGTH	0 -	seconds	0	1	A client exceeding the warning threshold will only be flagged as being in a warning state if it stays above the threshold for MINWARNINGLENGTH seconds. Not specifying or setting a value of 0 means any single sample above the warning threshold will constitute a violation.
ONLYREPORTPRIMARY	0, 1	NA	0	1	When set, only process issues on a client/source if that source is or may be primary at the time of the issue. Non-primary sources will not have threshold violations recorded. If the reporting client data doesn't indicate what source was primary, it's assumed that may be and will be processed. (TimeKeeper versions 7.2.14 and above include this detail for processing.)
STARTTIME	00:00:00 - 24:00:00	NA	NA	13:00:00	Specifies the offset from midnight (00:00:00 hours) UTC that the report will START coverage in its output. The default value indicates starting at midnight UTC.
TITLE	text	NA	NA	'London hosts'	A string naming a particular report.
WARNINGTHRESHOLD	0.0 -	seconds	(see Description)	0.000025	Clients that exceed this value will be flagged as in a warning state. This feature detects clients that may get near the alert threshold before the alert threshold is reached. Not specifying this value disables the feature.

Authentication - users and protocols

There are several login and authentication protocols supported by TimeKeeper. The web interface, when used on a TimeKeeper client, does not require local accounts on the system. On TimeKeeper Grandmasters, there are local system accounts that can be used for web, SSH and console logins. These and external accounts can be configured for remote authentication with RADIUS and TACACS+. This section and the "Grandmaster authentication - users and protocols" section cover the specifics of each of these types of logins and how they are authenticated.

Web interface authentication

For details on web authentication on TimeKeeper grandmasters, please refer to the section, "Web interface authentication." In this section, we'll cover web authentication on non grandmaster installations of TimeKeepers - clients, servers, boundary clocks, etc.

Admin web user

When the web interface is enabled, users can log in with an "admin" account, with the default password "timekeeper". Note: In earlier versions of TimeKeeper the default password was "fsmlabs". This account is internal to TimeKeeper and is not a local system (Windows/Linux) account that can be logged into. On Grandmasters, TimeKeeper has a local system account, detailed in the section, "Grandmaster authentication - users and protocols."

Logging in as *admin* will allow you to manage all aspects of TimeKeeper, including visualization, service management, configuration, and other supported options. For a more limited login, refer to the next section on the *readonly* user.

Readonly web user

A more limited user is also available with the *readonly* account that can be configured by the *admin* user via the web interface. The *readonly* user can log in and review TimeKeeper data, but cannot reconfigure or manage the system.

By default, the read-only user feature is present but not configured and cannot be used. To configure the readonly user, log in as *admin*. Select the *Configuration* tab, then select the *Service & System Management* subtab. The *Set readonly password* button will allow you to configure a password for the read-only user, which will be named *readonly*.

PAM authentication

To allow user logins with the PAM user name and password for Linux clients (non-Grandmaster installations), enable the ENABLE_PAM_AUTH parameter in the TimeKeeper configuration file. ENABLE_PAM_AUTH is supported only on Linux. Note that if the user is *admin* or *readonly*, they will not be authenticated via PAM. When the user successfully validates with PAM, they will have limited access, similar to the *readonly* account for the web interface. This feature enables LDAP authentication in the web interface on Linux.

TimeKeeper leverages PAM through the sshd config (typically /etc/pam.d/sshd) and will use this service for authentication. You can validate the *LDAP-authenticated* login using SSH as follows:

```
# ssh -o PasswordAuthentication=yes -o PreferredAuthentications=keyboard-interactive,password -o PubkeyAuthentication=no user@localhost
```

Issue this command on the TimeKeeper host that intends to allow LDAP-authenticated web logins.

NTP authentication

TimeKeeper supports NTP MD5 symmetric-key authentication. The path for the keys file on Linux is /etc/ntp/keys; on Windows, %ProgramData%\timekeeper\ntp.keys. The format of the keys file on Windows is the same as on Linux. On non-Grandmasters, it is up to the user to manage and secure this file.

To configure the keys on TimeKeeper Grandmasters, login as the admin user via keyboard/monitor, RS232 console, or ssh, and run timekeeper_cli. Select the "Configure NTP MD5 Keys" option to add or remove keys.

Grandmaster authentication - users and protocols

Here we'll go over authentication with TimeKeeper, with a specific focus on capabilities and configurations when TimeKeeper is installed on a grandmaster. We'll cover the system login details, RADIUS/TACACS+ authentication, ssh, and so on.

For details on TimeKeeper authentication abilities in the non-grandmaster case, please refer to the "Web interface authentication" section.

Web interface authentication

In this section, we'll cover web authentication specific to grandmaster installations of TimeKeepers.

Admin web user

When the web interface is enabled, users can log in with an "admin" account, with the default password "timekeeper". Note: In earlier versions of TimeKeeper the default password was "fsmlabs". This is a local system account, also usable via ssh and on the console. The password can be set via the GUI and also on the command line via ssh or on the console.

Logging in as *admin* will allow you to manage all aspects of the system, including visualization, service management, configuration, and other supported options. On grandmasters, the primary interface for controlling the appliance is via the web interface, so the admin user will be used for nearly all system management. For a more limited login, refer to the next section on the *readonly* user.

Authentication of the web login is handled based on how the unit is configured. By default authentication and password control is local to the system *admin* account. If RADIUS or TACACS+ is enabled, authentication is handled by the configured servers. Should access to these services fail (such as during a network outage), the provided password will be tested against the local system account.

For *admin* to be authenticated via RADIUS or TACACS+, the authentication server must have an account named *admin* with matching credentials.

If a TimeKeeper Grandmaster upgrade detects the need for and adds a local system *admin* account for authentication, it will as a precaution disable ssh access. This prevents unauthorized logins with the default password, until the *admin* account password can be changed to something unique. SSH access can then be reenabled as before via the web GUI.

Readonly web user

A more limited user is also available with the *readonly* account that can be configured by the *admin* user via the web interface. The *readonly* user can log in and review TimeKeeper data, but cannot reconfigure or manage the system.

By default, the read-only user feature is present but not configured and cannot be used. To configure the readonly user, log in as *admin*. Select the *Configuration* tab, then select the *Service & System Management* subtab. The *Set readonly password* button will allow you to configure a password for the read-only user, which will be named *readonly*.

Logging in via ssh, console, RS232

By default the Grandmaster does permit RS232 console login, and keyboard/monitor access for enabled accounts but not ssh access.

To enable ssh access login to the web interface as *admin* click the *Enable SSH* button in the *Configuration* tab, *Service & System Management* subtab. Once this is enabled, logs can be retrieved via SCP, SFTP and similar tools with the *loguser* or *admin* user. It is also possible to SSH into the device and run system monitor tools like 'top', 'ps' and other Linux programs.

Users *admin* and *readonly* can also be used on the console or via ssh in addition to the *loguser* user. *readonly* and *loguser* accounts must be enabled via the web interface before they can be used. This can be done in the web interface under the *Configuration* tab, in the *Service & System Management* subtab. This will permit RS232 and keyboard/monitor access but not ssh unless ssh access is enabled.

All console and ssh logins will authenticate via RADIUS or TACACS+ if the Grandmaster is configured to use those protocols.

It is strongly recommended that you change the default password for *loguser* immediately. The default is: *logaccess*. This applies even if RADIUS or TACACS+ is in use, as TimeKeeper can still fall back to login against the local account when logging in.

The *readonly* and *loguser* accounts are intended for easy log collection via ssh, so that logs can be archived as needed for audit trails.

Root access

Standard shell access as *root* is not normally permitted or recommended. Please do not enable *allow root access* via the web GUI unless asked to by FSMLabs personnel.

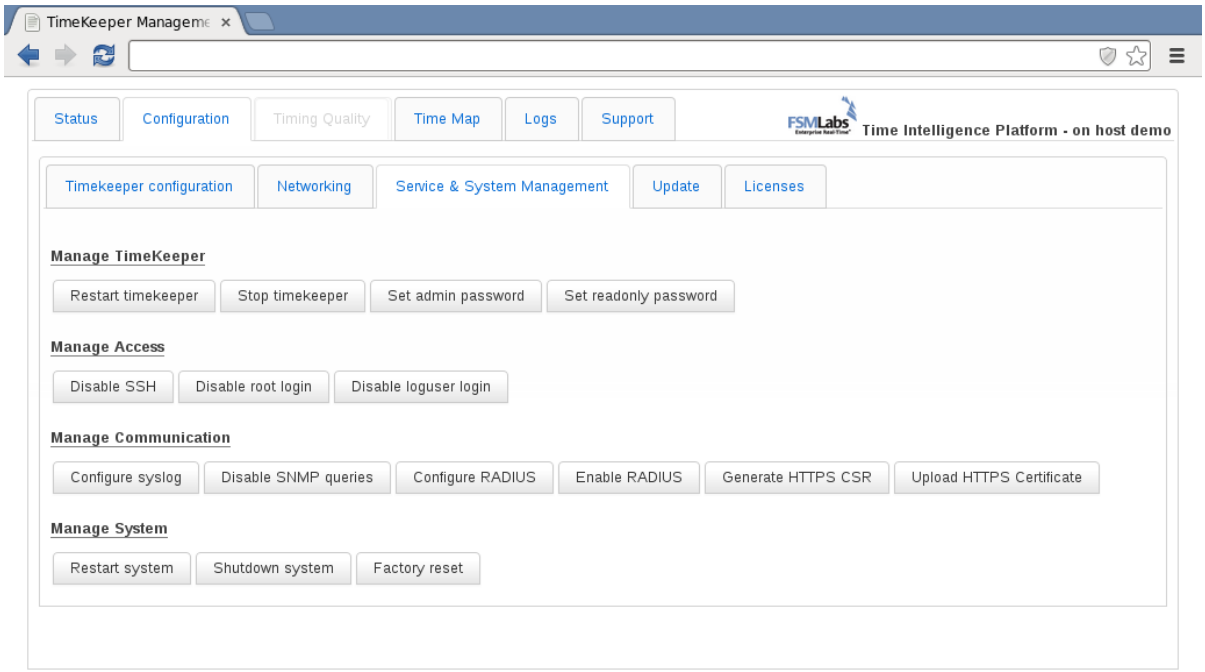
HTTPS support

The web interface on TimeKeeper Grandmasters may be accessed via normal http on a configurable port (see *WEB_MANAGEMENT_PORT*) or via https on port 443. If web management is enabled on a TimeKeeper Grandmaster, https will be available. If the *WEB_MANAGEMENT_IP* setting is used to restrict which interfaces respond to web queries, https support will also be limited to that named address. Access over https is achieved with the following example URL:

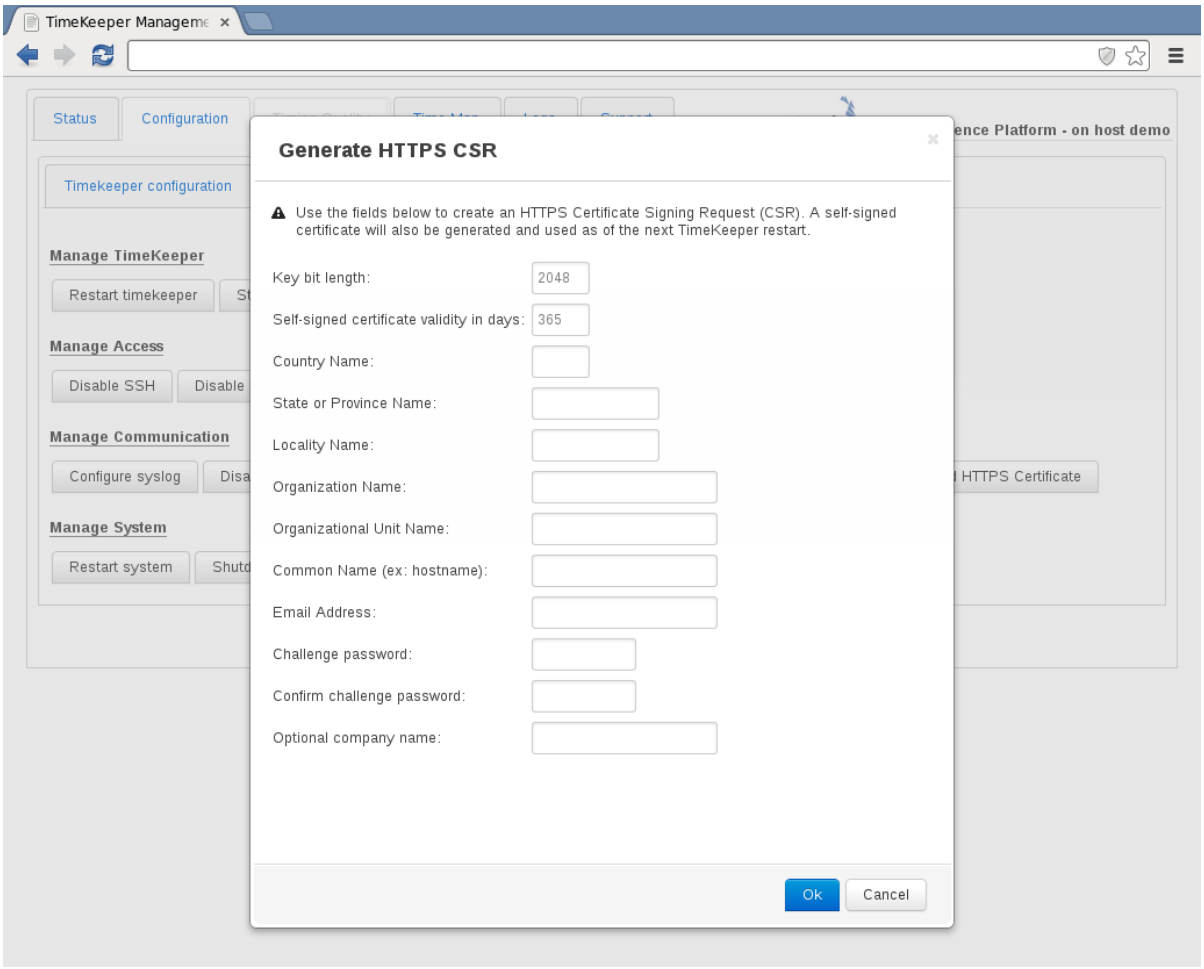
```
https://192.168.1.1
```

Out of the box, the grandmaster will use a unique self-signed certificate. This means that browsers may, on initial connection, ask the user to confirm that the certificate is acceptable. This is normal and expected. This warning will only occur when using https, and with most browsers will only need to be viewed once.

Users can configure https support by logging into the appliance over https. Navigate to the *Configuration* tab, subtab *Service & System Management*. Options to *Generate HTTPS CSR* and *Upload HTTPS Certificate* will be present. (Note that if the system is not a Grandmaster, or if the login was not done over https, these options will not be available.)



The existing self-signed certificate can be used, or users can provide their own certificate for the appliance. A new signed certificate may be needed to comply with internal policies, or it may just simplify systems management. Whatever the reason, a Certificate Signing Request can be made by clicking on *Generate HTTPS CSR*. This will prompt the user for the information needed to create a new request for their certificate authority. Enter the information, and wait while the request is generated.



Once the request is generated, the dialog will be updated with the CSR data. This can be copied and delivered to the appropriate certificate authority for validation. As part of this process, a new certificate and key will be generated on the appliance. The next time TimeKeeper is restarted, the new key will be used - but this process does not require an immediate restart. (Note that since the appliance has a new certificate, client browsers may present that same initial certificate warning after the next appliance restart.)

Once a new certificate has been generated based on the CSR, it can be uploaded with the *Upload HTTPS Certificate* option on the same tab. This is a file upload dialog that can be used to upload the newly generated certificate. This step will cause TimeKeeper to restart immediately to apply the certificate.

RADIUS/TACACS+ support

TimeKeeper Grandmasters support the use of RADIUS and TACACS+ for login authentication in addition to local accounts. (Note that, although this document simply refers to “authentication,” these servers provide authentication, authorization, and accounting capabilities.) To enable the feature, log into the web interface, and select *Configuration*, then *Service & System Management*. The option to enable/disable RADIUS and TACACS+ and to configure both will be visible under *Manage Communication* if supported. If these controls are not available, an upgrade may be required. Please contact support@fsmlabs.com for details.

By default, RADIUS and TACACS+ support is disabled. To enable it, first use *Configure RADIUS* or *Configure TACACS+* to configure for your authentication servers and pre-shared keys. A timeout value in seconds indicates how long to attempt to login via that server. If you have previously configured a pre-shared key for a server it will not be displayed if you reload the dialog, as a security precaution.

Once configured, logins on both the web interface, console, and SSH (if enabled) will attempt to authenticate via TACACS+ or RADIUS first, with each configured server in turn. If TimeKeeper is not able to access any of the RADIUS or TACACS+ servers, authentication will be attempted using the local account credentials.

This is very important - local account authentication will be used if TimeKeeper cannot access the RADIUS or TACACS+ servers. This means, at a minimum, local passwords must be changed from the default shipping settings. Accounts with local passwords include:

- loguser - local account, change password with *passwd* via console, SSH, or web GUI
- admin - local account, change password with *passwd* via console, SSH, or web GUI
- readonly - local account, change password with *passwd* via console, SSH, or web GUI
- root - local account, change with *passwd* via console or SSH

If TimeKeeper is configured to use TACACS+, and the configured server hosts are down or otherwise unreachable, it could take up to one minute to timeout per host. Therefore, if it seems like TimeKeeper is unresponsive when you attempt to log in (with either local or external credentials), you should wait a few minutes for a response before, for example, refreshing the web page or restarting the Grandmaster. To resolve this, make sure the configured hosts are reachable or, once you log in, disable TACACS+ and use local credentials and authentication until that happens.

If external authorization is required, either TACACS+ or RADIUS can be used, but the Grandmaster will not support both protocols simultaneously.

If the user logs into TimeKeeper with valid RADIUS/TACACS+ information, and there is no local account of that name, the user will be presented with 'readonly' or 'admin' views, detailed above.

WARNING - If you have been relying on credentials for an internal account, such as admin, while configured to use *external* authentication, you should create an external account with that same name and its own credentials and privilege attribute (see "Authorization" section below). If you don't, you will be unable to log in with that account because, starting with version 8.0.17, TimeKeeper no longer falls back to local authentication if external authentication fails, e.g., unknown account or bad password. (It now only falls back if the authentication servers are inaccessible, e.g., offline.) Also, as of 8.0.17, the local admin white list has been replaced with support for an administrative-privilege-level attribute on your authentication servers (see below). To maintain administrative access, you must add such an attribute for each name in that list on your authentication servers.

Authorization

Local authorization is tied to the account name. For example, the account with the admin username has administrative privilege level. The privilege level of an *external* account, created and maintained by users on their RADIUS and TACACS+ servers, is indicated by the priv-lvl attribute common to other network equipment. Allowed values are 0 through 15. Level 0 indicates read-only privilege level; level 15 indicates administrative privilege level. The read-only privilege level is also assigned if the level is any other value or the priv-lvl attribute is missing.

Accounts with administrative or read-only privilege level have the same capabilities as the local admin or readonly account, respectively. For ssh and console access, the external user starts in the home directory corresponding to the local account. While you can create an external account with the same name as a local account, i.e., admin, readonly, or loguser (but not root), it starts in the same home directory and has the same capabilities as the local account–you can't override this by specifying a contradictory privilege level.

RADIUS

For the RADIUS privilege-level attribute, the FSMLabs vendor has an id of 42733. The AVPair string attribute has an id of 1. The attribute value is of the form, shell:priv-lvl=N, where N is the privilege level, e.g., 0 or 15.

Here is a read-only example for the FreeRADIUS RADIUS server:

```
jsmith Cleartext-Password := "$kZ90mhvSm41"
       FSMLabs-AVPair = "shell:priv-lvl=0"
```

TACACS+

The TACACS+ privilege-level attribute is of the form, priv-lvl=N, where N is the privilege level, e.g., 0 or 15.

Here is an administrative example for the tac_plus TACACS+ server:

```
user = jsmith {
        global = cleartext "$kZ90mhvSm41"
        service = exec {
                priv-lvl = 15
        }
}
```

TimeKeeper web interface

We covered the basic features of the web interface in the "Web management"" section, but here we'll look at a couple of technical specifics.

The web interface will visualize a number of things for you, including:

- Current status
- Configuration
- Time accuracy, network delay, frequency behavior for sources
- Timing client behavior
- A timing map of your network
- Compliance reports about your clients

and more. It also provides an easy way to collect logs for TimeKeeper support. The web interface is available on TimeKeeper clients, servers, and grandmasters. Depending on the license, more or fewer features may be visible, and on grandmasters where there are more hardware features available, there will be more options for management and control.

On TimeKeeper grandmasters, the web interface is also reachable via HTTPS in addition to HTTP automatically. For TimeKeeper clients and other installations HTTPS is enabled by default, but HTTP is disabled (WEB_MANAGEMENT_PORT=-1). These non grandmaster installations now include a self-signed certificate.

The web interface is controlled with 2 primary options, enabling the option and controlling the port it's listening on:

```
ENABLE_WEB_MANAGEMENT=1
WEB_MANAGEMENT_PORT=-1
```

HTTPS support is enabled by default and will serve the web interface via HTTPS on port 443. By default the WEB_MANAGEMENT_PORT is set to -1 so that HTTP is disabled.

Configuring for HTTPS

HTTPS support is provided out of the box on TimeKeeper Grandmasters and other installations like clients, Compliance installations, servers, boundary clocks, and so on. These installations will use a self-signed certificate which means that browsers may, on initial connection, ask the user to confirm that the certificate is acceptable. This is normal and expected. This warning will only occur when using HTTPS, and with most browsers will only need to be accepted once. You can provide your own certificate and key using the instructions below to avoid these warnings.

Configuration of this key and certificate are up to the user to manage as the particular commands and tools used to create the files may vary depending on the host. These files must be in the *management/* directory of the TimeKeeper installation:

- cert.key (key file for the certificate)
- cert.crt (HTTPS certificate file)

As an example, here are some possible commands used to create an HTTPS key and certificate:

```
# openssl genrsa -out /opt/timekeeper/management/cert.key 2048
# yes '' | /usr/bin/openssl req -new -key /opt/timekeeper/management/cert.key -out /tmp/cert.csr
# openssl x509 -req -sha256 -days 365 -signkey /opt/timekeeper/management/cert.key -in /tmp/cert.csr -out /opt/timekeeper/management/cert.crt
# chmod 600 /opt/timekeeper/management/cert.{crt,key}
```

A more stringent procedure likely applies in a production environment, this is provided for demonstration purposes only.

It's important to restrict access to the generated files, whether they're created directly as above or generated elsewhere and put in place manually. Above on Linux chmod is applicable, on Windows an ACL may be more appropriate, the specifics are up to the user to decide.

If a certificate and key exist when upgrading TimeKeeper, they'll be retained and reused after the upgrade completes. When building a TimeKeeper

RPM, the key and certificate is retained in the *timekeeper-config* package. See the timekeeper-config.spec file for more details.

To disable HTTPS, remove the certificate and/or the key. Without those files TimeKeeper will not provide HTTPS support, but will still serve HTTP on the configured port.

This table summarizes the configuration options for HTTP and HTTPS:

	ENABLE_WEB_MANAGEMENT	WEB_MANAGEMENT_PORT	management/cert.*
neither	0	NA	NA
HTTP only	1	0..65535	absent
HTTPS only	1	-1	present
HTTP & HTTPS	1	0..65535	present

(Because HTTPS is supported out of the box on TimeKeeper Grandmasters and cannot be disabled, the last column does not apply to them.)

If the private key in use requires a passphrase, that passphrase must be added (in encrypted form) to the TimeKeeper configuration using the HTTPS_KEY_PASSPHRASE option. To encrypt the password for inclusion use the *encodepassphrase* tool provided with TimeKeeper. For example:

```
# /opt/timekeeper/release64/encodepassphrase
Enter passphrase: fsmlabs
Please add: 'QCphe+KSxRK/PgGUOstVzQ==' as the value for the appropriate
configuration option in your timekeeper.conf file.
Restart TimeKeeper to apply the change.
```

would result in this line being added to timekeeper.conf:

```
HTTPS_KEY_PASSPHRASE='QCphe+KSxRK/PgGUOstVzQ=='
```

Restricting web access to a particular IP address

There is also a *WEB_MANAGEMENT_IP* option that may be specified as follows:

```
WEB_MANAGEMENT_IP=10.10.10.1
```

This will configure the web management tools to only listen on a specific IP address on the server. If web tools are enabled but this option is left unspecified, browsers will be able to connect from any interface.

Managing the TimeKeeper service

Once installed TimeKeeper can be started during system boot time like any other system service with the *ntsysv*, *update-rc.d*, *chkconfig*, or *systemctl* commands, or via the TimeKeeper web interface. On Windows it can be managed via the Windows Service manager or with the *sc* command.

Once installed, TimeKeeper is listed as the service “timekeeper”. By default it is not configured to start on boot and must be manually enabled. When started on Linux, TimeKeeper will stop services that conflict with proper operation, such as ntpd. If ntpd is running when TimeKeeper is started the ntpd service will be shut down. TimeKeeper will not reconfigure the NTP service to not start at system boot.

To start TimeKeeper by hand on sysvinit-based Linux systems:

```
# bash /etc/init.d/timekeeper start
```

Similarly, *stop*, *status*, or *restart* can be used in place of *start* above to stop, get the status of, or restart the TimeKeeper service, respectively.

If your system is based on systemd, you can start/restart/stop TimeKeeper similarly:

```
# systemctl start timekeeper
```

With systemd-based distributions, TimeKeeper also depends on *network-online.target* so that interfaces and routes are up and configured before TimeKeeper starts.

On Windows TimeKeeper can be started from an elevated command window with:

```
c:\> sc start timekeeper
```

Substituting *stop* for start will stop the TimeKeeper service. The service can also be managed with the normal Windows Service Manager (“services.msc”).

Verify TimeKeeper is working

Within a couple of minutes TimeKeeper should be active and running if the time source is configured properly. You can verify this by running the command *tkstatus* on Linux, *tkstatus.bat* on Windows. This will report if TimeKeeper is active or not, what the current clock offset is and how old the latest timestamp is along with the same data about all configured sources. Run *tkstatus -f* on Linux, *tkstatus.bat -f* on Windows to report the one-way delay in addition to the base information.

If you’ve enabled the web interface, you can login to TimeKeeper via the web (port 8080 by default) (default passwords are in the “Web interface authentication” section). The *Status* and *Timing Quality* tabs on the web interface also provide a clear summary of which source is being used, the current accuracy, and also details on the behavior of the various configured time sources over time.

Also, you can refer to the log files directly. */var/log/timekeeper_0.data* (again, located under the TimeKeeper installation directory on Windows) is a running list of every update that TimeKeeper receives and how TimeKeeper is adjusting time for source 0. This file contains a line for every update received from the time source. If it is growing then TimeKeeper is receiving time updates.

In the file, column 1 is the absolute UTC timestamp from the time source in seconds since midnight January 1 1970. Column 2 is the estimated local time offset as of that update. The second column gives a good indication as to the quality of the sync that TimeKeeper has, and will converge shortly after TimeKeeper starts.

For more details on the log file contents, refer to the “File formats” section.

If this file is not being updated, it's likely that firewalls are preventing receipt of network data, or there is some configuration issue reported in */var/log/timekeeper*.

Log rotation

TimeKeeper is provided with two default logrotate recipes that rotate all of the relevant TimeKeeper log files. On Linux this gets installed in /etc/logrotate.d/. By default, this will retain 7 days worth of logfiles. If you have specific log storage needs this value can be easily modified directly or, on the TimeKeeper Grandmaster appliance, via the TimeKeeper web interface.

TimeKeeper's logrotate configuration options were updated as of 7.2.13. When downgrading from a 7.2.13 or later release to a pre-7.2.13 release, be sure to check logrotate settings because this can result in unexpected values.

Log locations

TimeKeeper logs information in multiple files, stored by default at these locations:

- Linux: /var/log/
- Windows: %ProgramFiles%\timekeeper\var\log

The default location can be overridden with the LOGDIR configuration option. A full path without spaces must be specified. To use the Windows default value, do not specify anything for the LOGDIR configuration option.

Files stored here include:

- timekeeper (timekeeper.log on Windows) - central place for logging alarms, general messages, state change information and administrative information
- timekeeper_$SOURCE.data - one file per configured source
- timekeeperclients/ - directory of per-host and per-source data files for each reporting client

Each source in timekeeper.conf will have a timekeeper_$SOURCE.data file that corresponds to the configured source number. The file contains sync information about that source over time. Similarly, each client file in the timekeeperclients/ directory will have sync information about each client's source over time. All sources have the same log format regardless of source type.

File formats

Every update that TimeKeeper receives for a particular source is recorded in the relevant source file, one line per update. The data files for each client and each client's source stored in *timekeeperclients* does the same, with one update per client sample, with the same data format in both locations.

The best way to visualize and understand the behavior is to refer to the "Visualizing data" section. The data is an open format so that users can work with the data with the tools they need. The log file format is as follows -

In most installations, the first two fields - the absolute timestamp and the offset from the time source - are all that is of interest. (A positive offset means the system clock is ahead of the source clock; negative, the system clock is behind the source clock.) Once started and running properly this file should be growing as updates are received.

- Column 1: Absolute timestamp of the update, seconds since midnight January 1 1970 (also known as UNIX time)
- Column 2: Offset (in seconds) from the remote server or time source
- Column 3: Raw offset from remote side - pre-smoothing and adjustments
- Column 4: Internal TimeKeeper use
- Column 5: Clock correction factor for the TimeKeeper managed clock
- Column 6: Used to provide additional data with TimeKeeper GPS devices
- Column 7: Used to provide additional data with TimeKeeper GPS devices
- Column 8: Internal TimeKeeper use
- Column 9: Timestamp type - hardware/software/other timestamp source
- Column 10: Network delay - meaning depends on type of source
- Column 11: Raw network delay - meaning depends on type of source (for NTP/PTP this is the one-way trip time)
- Column 12: Time source, PTP, NTP, PPS or other
- Column 13: "Ideal" clock correction that if used would have minimized offsets over the previous few minutes
- Column 14: Model error over the model interval period
- Column 15: Source traceability information, see below.
- Column 16: Internal TimeKeeper use
- Column 17: Reported upstream source accuracy, if available
- Column 18: Internal TimeKeeper use
- Column 19: Internal TimeKeeper use

If this file is not being updated, it's likely that firewalls are preventing receipt of network data, or there is some configuration issue reported in the *timekeeper* file.

The amount of detail available in each file will vary depending on the client. For some remote client types, the data available is limited. If the client is a TimeKeeper client, the data provided in that file will be more fully populated, giving you a better understanding of timing quality across the network.

Column 15 options include "PRI", "AUX", "MON", "INV", "UTC" and "HOL". For reporting purposes, if this field includes the keyword "PRI", it is the

primary source by which the system's clock is being steered, otherwise it is marked “AUX”. “UTC” indicates that the source reports it is traceable to UTC. “MON” indicates that the source is marked MONITORONLY. “INV” indicates that sourcecheck has marked the source invalid. “HOL” denotes that a time source is in holdover.

Visualizing data

The best tool for visualizing TimeKeeper's data is the web interface, covered in the "Web management" and "TimeKeeper web interface" sections, or with the PDF reports generated by TimeKeeper Compliance.

plot.sh

However, in some cases it's helpful to post process data, either on the host directly or on another machine. A script is provided for this on Linux, and can interactively plot timing data or generate plot images of aspects of timing quality over time.

The script is provided here:

> /opt/timekeeper/release64/plots/plot.sh

Also included in that directory are a number of gnuplot scripts to parse TimeKeeper files. *plot.sh* can use any of these, but *offsetplot.in* is of the most interest to users (this is the default plot). It shows the clock offset from the source, both the raw input TimeKeeper received, and the smoothed clock presented to the user.

Here's an example of how to use the script:

```
# ./plot.sh
```

or for a specific file:

```
# ./plot.sh -f /var/log/timekeeper_1.data
```

This will show an interactive plot of the local system's TimeKeeper data, updating every 5 seconds. Similarly:

```
# ./plot.sh -l user@host:/var/log/timekeeper_0.data -f 30
```

Will plot results of TimeKeeper data from host 'host', retrieving the data file and updating every 30 seconds. Both of the above commands will generate plots of the clock offset - specifying *-p plotfile.in* allows you to create a different graph. Another example is:

```
# ./plot.sh -p clockrateplot.in
```

This will generate a plot of the changes in clock rate on the local machine running TimeKeeper. Running *plot.sh -h* will print a full list of accepted arguments.

The gnuplot scripts can also be customized to display other information as needed.

Extracting data with tkcat

tkcat is a command line tool to extract TimeKeeper source data from the log files. It's handy to use when you want timing behavior about a particular source over a specific span of time. That data might be in the most recent .data file, or it may be in a previous logrotated file, or it may span multiple files. tkcat finds and extracts that data for you without having to worry about which files contain what data.

If you wanted to get data from source 1 about its behavior from 6am-4pm June 1 2018 (UTC), you'd get the start and stop UTC epoch seconds for that time window (1527832800 and 1527868800 respectively), and pass that information along to tkcat, which is provided in the release64 directory within the TimeKeeper installation:

```
# ./tkcat -e 1527832800 -l 1527868800 -f /var/log/timekeeper_1.data
```

This will print to stdout source 1's samples from within that time period. If the samples span logrotated or compressed files that's fine, tkcat will handle that detail and just provide the clock sample data. This can be redirected and then optionally fed into other tools like plot.sh above, Excel, and so on.

Other options include the ability to subsample with -s (only provide 1000 samples over the given time period), -r for human readable output, and -n to only extract data from the newest file.

Alternative tools

Since the data is stored in an open format, documented in the "File formats" section, there are other alternatives for post processing data too.

The data is easily imported into Excel, at which point many common processing and plotting options are available.

Leap seconds

TimeKeeper handles leap seconds by “slewing” in the default mode and will not introduce discontinuities, “jumps” or pauses in time to correct them. It will smoothly adjust any offset introduced by a leap second to ensure monotonically increasing time. Some time keeping methods and time reference sources do pause the clock and TimeKeeper will deal with them gracefully. This will initially show up as about a 1.0 second offset of the time which will be smoothly corrected out over the next three to eight minutes.

TimeKeeper does offer a mode to “jump” the clock or perform a “step” when the leap second occurs in order to reduce the amount of time needed to correct out the time offset introduced by the leap second. This can be enabled by setting STEP_ON_LEAPSECOND=1. When TimeKeeper detects a 1 second offset within a 15 second window of when leap seconds are introduced (either midnight UTC of June 30 or December 31 of a year) it will immediately correct the clock by 1 second. Depending on how leap seconds are propagated within your time environment this may not occur exactly when the leap second occurs.

We recommend testing how a leap second is handled in your specific environment by simulating one if very fast correction of a leap second is important to you. Feel free to contact support@fsmlabs.com if you would like assistance.

TimeKeeper using external reference devices

Please refer to the documentation for your time reference device for specifics on how it handles leap seconds since this will determine how TimeKeeper responds.

If a direct GPS or other absolute time source is being used that provides time via a UTC broken down format then normally an additional second will be sent to TimeKeeper as the 23rd hour, 59th minute and 60th second of the day when the leap second happens. This will show up on the server as 1 second after the 23rd hour, 59th minute and 59th second of the day. When the new day starts in the next second - 0th hour, 0th minute and 0th second TimeKeeper will calculate this as the same second as was previously reported by the time source. This will mean TimeKeeper will calculate an error of 1 second at that time.

Over the next few seconds TimeKeeper will work to remove that error through slewing. Normally this takes 5 to 10 seconds to do. It should be noted that an offset will be expected when leap seconds are introduced and will take 5 to 10 seconds to eliminate. During the time that this offset is being reduced TimeKeeper will not slow down the clock by more than 25% of its original value.

TimeKeeper as an NTP client/server

Most NTP protocol implementations will handle leap seconds by pausing the clock at some point in the last second of the day. TimeKeeper does not do this. Below is an example of querying a typical non-TimeKeeper NTP server at 0.25 second intervals during a leap second.

```
23:59:58.50
23:59:58.75
23:59:59.00
23:59:59.25
23:59:59.50
23:59:59.75
23:59:59.75
23:59:59.75
23:59:59.75
23:59:59.75
00:00:00.00
00:00:00.25
```

When TimeKeeper is acting as a client to one of these non-TimeKeeper NTP servers TimeKeeper will slow it’s own clock down slightly so that it can match the remote clock that is paused until the 1 second offset that was introduced by the leap second is corrected. This offset removal should take a few seconds.

TimeKeeper acting as a NTP server will set the ``leap second indicator'' flag in responses that it sends. It will track the time of any source that it is

configured to use and will send that time out to any requesting NTP clients.

TimeKeeper as a PTP client/server

TimeKeeper as a PTP client will see a leap second occurs when a PTP Announce message sent by the server shows a change in the UTC offset field. At that point TimeKeeper will begin to correct the offset introduced by the change in TAI to UTC offset values. This occurs whenever the server sends the new UTC offset in an announce message and that can vary depending on the server. This could be quite some time after the actual leap second.

TimeKeeper acting as a PTP server will set the “leap second indicator” bit in announce messages to clients. When the leap second occurs according to the source feeding the server TimeKeeper will slew to match the source and will provide that time to the client, unless configured to step as described above.

Tools included

TimeKeeper installs with a number of tools, some covered elsewhere in this document. Many visualization capabilities are handled by the provided web interface and plotting tools covered in the section, "Visualizing data."

Other helpful tools installed in timekeeper/release64 may include:

- baddate - This is a (Linux) test utility for moving the time away from the current value. The argument to baddate is a floating point number of seconds to move time either forwards or backwards. This can be used before TimeKeeper starts, in order to move the system time around for testing.
- timekeeper_uninstaller, timekeeper_uninstaller.bat - This is an uninstall script to remove a TimeKeeper installation. Use timekeeper_uninstaller.bat on Windows.
- report_problem.sh and report_problem.bat (also available on Linux as /usr/bin/report_problem.sh) - This script collects system information to be emailed in case of a problem. Please provide the results of this script with the description of any product issue. Running *report_problem.sh -u* (*report_problem.bat -u* on Windows as Administrator) will try to automatically upload the generated report to FSMLabs. The -d option may also be used to specify a writable directory for report generation. Both of these scripts can be more easily run from the *Help* tab on TimeKeeper's web interface.
- tkstatus.sh (also available as /usr/bin/tkstatus on Linux) - This script gives a short overview of the current TimeKeeper status, including a license overview, current source accuracy, and the age of any recent updates. On Windows tkstatus.bat is available in the release64 directory in the TimeKeeper installation. Running *tkstatus -f* (*tkstatus.bat -f* on Windows) will give a full, untruncated overview of the current TimeKeeper status, including all of the previous fields and one-way delay.
- stats - This Linux tool summarizes TimeKeeper source files to give offset and frequency data. It can be run as "/opt/timekeeper/release64/stats timekeeperdata=/var/log/timekeeper_0.data" to provide a summary of the primary source's behavior over time.
- timekeeper_cli - This is a tool that allows for some command-line based configuration ability on TimeKeeper Grandmasters.
- encodepassphrase - This tool allows for encryption of passphrase settings, needed for either SNMPv3 alerts or to access key files to provide HTTPS support.

Tips

Collected here are some short notes about various aspects of testing, drivers, and general practices. If you have any specific questions about any of these items, please contact support@fsmlabs.com.

Assumptions about time in the past being invalid

When receiving time from any time source TimeKeeper will validate that time to make sure that it can be trusted. One of these checks includes making sure the time is not earlier than the release date of the TimeKeeper version being used. This can catch problems with time sources that are reporting erroneous time but it also means that it is not possible to run simulations with time in the past. If you need to be able to use simulated time that is in the past please contact support@fsmlabs.com and we can explain how to configure TimeKeeper to allow that.

Holdover

Some time devices are capable of keeping good time while not actively receiving time updates. This includes the Spectracom TSync GPS/PPS card, Symmetricom BC637 GPS/PPS card and the TimeKeeper GPS/Oscillator. By default TimeKeeper will remain in holdover for 2 hours (7200 seconds) and continue to use the time source even when it reports no GPS/PPS signal. That time limit can be changed with the *HOLDOVER_LIMIT=X* setting where *X* is number of seconds. Once that time limit expires TimeKeeper will begin comparing all time sources to determine if the holdover time is out of range with other time sources by using the “Sourcecheck” TimeKeeper algorithm (even if Sourcecheck is disabled).

Clock adjustment and steering

TimeKeeper will smoothly adjust the clock to correct offsets in most cases. However, when the offset is greater than 5 seconds TimeKeeper will reset the time to avoid the delay necessary to slew the clock. Once the clock is set in this way it will continue tracking and slewing as necessary to keep the time synchronized. In practice the only time this happens is when TimeKeeper is first started and it begins synchronizing the clock with a remote time source.

Once TimeKeeper is active, setting the system time is not recommended. TimeKeeper will stop other timing daemons but it is important that other utilities do not also try to steer the clock. That includes the command line tool “date” and other system tools that set the time and time synchronization software such as ntpd that use system calls such as “settimeofday” and “adjtime”.

Windows

The Windows Time service, W32Time, includes an NTP daemon. It can cause problems in a couple of ways. First, it could be trying to steer the system clock in conflict with TimeKeeper. Second, even if it isn’t steering the clock, it could be bound to the same port that TimeKeeper uses. The following instructions are just a starting point for dealing with these issues. You should research Windows Time service options to determine what best suites your needs.

You can stop W32Time and prevent it from automatically starting on boot with these two commands:

```
sc stop w32time
w32tm /unregister
```

However, another process, e.g., the Windows Domain Controller, could start it up again, so it’s up to you to assure that this can’t happen. If this becomes a problem, e.g., you require Windows Domain Controller, you can leave W32Time running but prevent it from steering the clock with a registry setting (set Type to NoSync). There are three ways to do this.

1. You could use this command:

```
reg add HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\W32Time\Parameters /t REG_SZ /v Type /d NoSync /f
```

2. use regedit to make the same changes,
3. or use this command:

```
w32tm /config /syncfromflags:NO /update
```

After making any changes to the W32Time configuration, you should restart it with this command:

```
net stop w32time && net start w32time
```

If you have TimeKeeper configured to use NTP, there could still be a problem. W32Time could be bound to the NTP port even though it's not using it. In that case, TimeKeeper cannot use NTP sources. Configure TimeKeeper to use something else instead, such as PTP.

Identifying inaccurate sources

Accuracy issues may manifest as a noisy primary clock - usually as noise in the smoothed offset plot on the web interface, or in the second column of data in timekeeper_0.data.

Comparing the noisy data file to a less noisy one can help identify where the noise is coming from. If your primary source is SOURCE0, and it appears to be smoothly handled in *timekeeper_0.data", and SOURCE1 is noisy in the web interface and *timekeeper_1.data", you can be fairly certain that either the network or the time server itself in SOURCE1 is noisy.

This is worth noting because if the noisy source is your primary and TimeKeeper is chasing that clock, the reported offsets for your secondary and other sources may appear to be noisy also. Put another way, if your SOURCE0 is noisy but SOURCE1 is stable, chasing the noise in SOURCE0 can appear in some situations to also be noise in SOURCE1.

Where it's an option, configuring your primary source to be a clean and stable PPS can quickly indicate which sources are actually noisy. When TimeKeeper is not chasing a noisy clock and is instead tracking a stable PPS, much of the noise will be removed, allowing for more accurate measurements.

Sync rate

The default sync rate of 1 and 0.9 packets/second for PTP and NTP, respectively, is a good tradeoff between network traffic, CPU load, and synchronization accuracy. A higher sync rate won't necessarily increase accuracy. If you're considering it, experiment first in your environment to determine if it actually results in better accuracy than with the default. Also note that higher sync rates increase storage requirements for Compliance reporting. Please refer to the Compliance "Disk space requirements" section.

Unresponsive NTP server

When acting as a NTP client TimeKeeper will wait for many milliseconds for a response. If there is no response from the server in that time the server is assumed to not be responding and no time synchronization will occur during that query of the server. This could be a problem for synchronization of a very slow WAN connection. If your environment and network have a very long round-trip between the client and server please contact support@fsmlabs.com for configuration information.

Hardware timestamping network cards

Hardware timestamp support allows TimeKeeper to most accurately measure receipt and transmission of timing data. This is supported automatically if:

- The network hardware supports timestamping

- The driver for the network hardware can enable the feature
- The host distribution supports the calls that enable the feature
- TimeKeeper finds the reported data to be accurate

Hardware timestamping is supported on these cards (but not limited to them):

- Intel 82576, 82580, I350, I210 cards
- Mellanox CX-3, CX-4, CX-5
- Most Solarflare cards
- Broadcom 5719 cards

In the case of Red Hat Enterprise Linux 6, upgrade the provided driver with the most recent IGB version available from Intel since the driver shipped with the distribution does not support all cards. Version 2.4.8 of the IGB driver or above is recommended. Some version of these drivers do need to be loaded, configured with an IP address to initialize them (with *ifconfig*), unloaded and then loaded once again for proper behavior when used for hardware timestamp assist.

Hardware clock

TimeKeeper also updates the hardware clock every 10 minutes on Linux to keep it in sync with the current system time. This ensures there's a good reference for any components that depend on the hardware being in sync with the system clock.

Identifying accuracy problems

Poor NTP implementations

Many NTP implementations are of very low quality and are very noisy, even if the network delivering the NTP data is fast and deterministic. In these cases it's recommended that TimeKeeper be configured to minimize the effect that these bad NTP servers can have on your network. This can be accomplished in several ways:

- Replace the NTP server entirely and distribute NTP with TimeKeeper to get accurate timing data to your clients
- Place a TimeKeeper server between your clients and the existing server, acting as a stratum server. In this configuration, TimeKeeper accepts the noisy NTP and smooths it out in order to serve it directly to clients
- On the clients, specify LOWQUALITY=1 in the sources that use the noisy NTP server. This way TimeKeeper will perform additional smoothing to the noisy clock to extract more signal

Firewalls

Although uncommon, occasionally firewalls are used between timing clients and servers. While the clients can still sync through the firewall, generally this reduces the potential accuracy of the client greatly.

Timestamping accuracy

TimeKeeper will accurately serve or consume time on any supported host, but there are some configurations that provide more accurate results.

TimeKeeper will enable the highest resolution timestamping it can, and report which features were enabled in */var/log/timekeeper* at startup. On windows, this will be logged to *%ProgramFiles%\timekeeper\var\log\timekeeper.log*. If the host distribution supports the feature, the driver can enable hardware timestamping, and the device supports the feature, timestamping will be done by the card. If the card or the driver can't support the feature, software timestamps will be requested from the network stack, if the host can provide them.

Hardware timestamps are the most accurate, followed by software timestamps. TimeKeeper uses this data to factor out propagation delays in incoming time, and also to provide more accurate data when delivering time.

It is important to understand TimeKeeper does not need special support for your particular hardware - it will find and use the most accurate timestamping available, automatically, to the best ability of your host distribution, driver, and hardware.

If you have any questions about how to make sure you're getting the best performance in your environment, please ask *support@fsmlabs.com*.

Bonding is supported, including the ability to use hardware timestamps with the devices in the bond. However, care should be taken to make sure that the underlying devices are of the same device type so that there's a consistent feature set and performance across the members of the bond.

Best Practices

Here is a brief list of TimeKeeper best practices. These may not apply to all deployments, but they apply to many that depend on TimeKeeper for accurate and resilient timing services.

- Have multiple sources for both clients and Grandmasters, including backing the Grandmaster GNSS (they can fail, too).
- Have a mix of sources, e.g., PTP and NTP, to guard against protocol failure.
- Unless you have extraordinary needs, use the default sync rate because a higher rate does not necessarily mean better accuracy.
- Only use TimeKeeper's Sourcecheck feature for a set of heterogeneous sources, e.g,. sources reached through different paths, different protocols, different Grandmasters.
- Disable all verbose-logging settings in production when not investigating an issue.
- Client
  - Disable all other time daemons, e.g., ntpd and W32Time, and assure that no process restarts them.
  - Make sure TimeKeeper is not pinned to a CPU with already high load.
- Grandmaster
  - Use a GNSS receiver at the Grandmaster.
  - Have peer Grandmasters use each other as a source for failover, but, to avoid a common point of failure, make sure they don't share a GNSS receiver antenna.
- PTP
  - Disable PTP features in PTP-aware switches.
  - We generally recommend hybrid mode, but, for long PTP paths, where multicast is not propagated or not propagated efficiently, use telecom mode (all traffic is unicast).
  - TimeKeeper supports the BMCA but can usually provide better failover by, instead, configuring it to use multiple sources and, in some instances, enabling Sourcecheck.
- NTP
  - Use public NTP servers with care, such as when pointing to a NIST server for regulatory compliance, because they can vary in accuracy and availability.
- Alerting
  - Handle alerts by specifying an email address, syslog server, or SNMP manager.
  - Specify sync-error thresholds for every source according to your accuracy expectations so that you are alerted when it's exceeded.
  - Similarly, for *serving* time, specify sync-error thresholds per server so that you are notified when a client exceeds the threshold.
- Compliance
  - Create reports with alert/warning thresholds *more restrictive* than needed for regulatory reporting to identify hosts with marginal sync accuracy.
- Networking
  - Use network interfaces that support hardware timestamping.
  - For failover at the network interface, use active/backup bonding and/or access multiple sources through different interfaces.
  - Use the same speed on both ends of a connection, e.g., 10G and 10G, to avoid asymmetric delays.
  - Avoid busy or intermittently congested networks for timing data. A dedicated timing network is best.
- VMs
  - Determine if the hypervisor should set the guest VM's clock at startup.
  - Prevent VM integrations from steering the guest clock.
  - Minimize migration, such as via vMotion. Otherwise, your applications could have to cope with sudden jumps in time.

IEEE 1588 (PTP) tips

A full description of the IEEE 1588 (PTP) protocol is beyond the scope of TimeKeeper documentation but these are some important suggestions for configuration of PTP with TimeKeeper.

PTP modes/profiles

There are several flavors of PTP (called profiles). Below is an overview:

Full multicast (TimeKeeper's default mode)

In this profile the grandmaster sends out multicast messages (sync, followup and announce), clients send multicast delay requests, the grandmaster responds to each request with a multicast response. This is pretty wasteful of network and CPU resources since every client/grandmaster/switch has to handle multicast packets not meant for them.

Hybrid (what we generally recommend)

The grandmaster sends multicast messages (sync, followup, and announce), clients send unicast delay requests, the grandmaster responds with unicast responses. This is the most efficient since only systems that need a given packet have to see it and multicast on the network is kept very low.

Full unicast (also called Telecom Profile)

Each client sends a lease request to the gm, which it responds to in order to maintain a 15 minute 'lease'. The grandmaster sends unicast messages to each client and the full exchange happens unicast (delay request, response, etc). Clients and grandmasters have to re-establish the lease every 15 minutes and the grandmaster has to maintain a list of all the clients, their leases and so on. There is a great deal of memory and CPU overhead for this mode.

Delay measurement mechanism

There are two mechanisms for measuring network delay.

The end to end delay measurement mechanism, or E2E, measures the delay from the grandmaster to the client. This can be used with network equipment that is not PTP aware. It calculates the network delay across the entire path between the server and client.

The peer to peer delay measurement mechanism, or P2P, measures the delay between each device on the network. This requires that all network equipment (switches, routers etc.) in the path be PTP aware.

The PTP delay measurement mechanism is controlled by the PTP_P2P configuration option and only applies to the client because the server responds to whatever request mechanism it sees (E2E or P2P).

Boundary Clocks

We generally recommend against using boundary clocks.

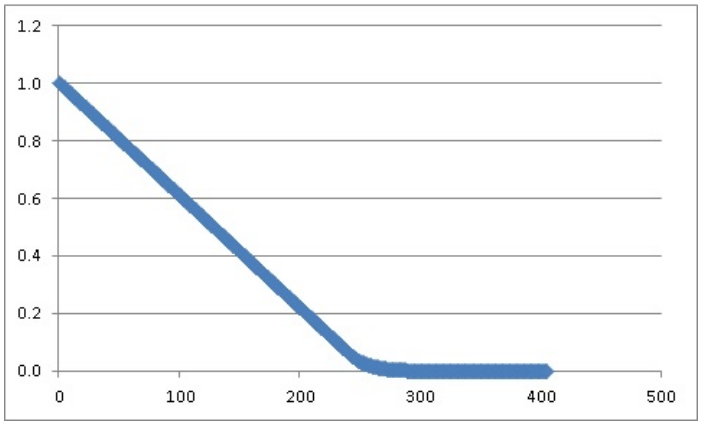
The first reason for this is that most boundary clock implementations (on switches) are buggy and introduce errors, connection problems and inaccuracies. In most cases we find that PTP problems where boundary clocks are involved are a result of the boundary clocks and one of the first suggestions FSMLabs support will make is to disable the boundary clock to see if problems continue.

The second reason is that most boundary clocks block PTP management messages. This prevents the exchange of accuracy information between PTP entities. That means recording the accuracy of systems in one place and generating reports on them isn't possible.

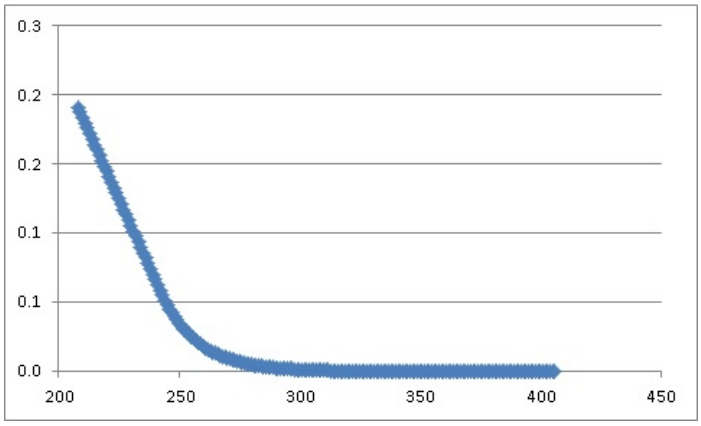
Slew rate

In most cases TimeKeeper will adjust the time by slewing. This means it will slowly reduce the time error to avoid erratic changes in time and also prevent erratic changes in the rate of time correction. This correction rate will never be greater than 1 second correction in 256 seconds or roughly 3.9 milliseconds correction per second.

As the time error gets smaller this rate will be reduced to asymptotically approach zero error. This means a gradual adjustment of time and also a gradual adjustment of the rate of time on the computer. This is illustrated below with an example of correcting a 1 second error.



The graph below shows more detail of the final stages of correction as the change rate is reduced. Note that the X axis is the error (begins at roughly 0.2 seconds). The rate of correction is gradually reduced as the error decreases providing a smooth transition to the correct time.



In most cases the correction of 1 second offset will require roughly 420 seconds or about 7 minutes.

Low Quality

The LOWQUALITY=1 setting instructs TimeKeeper to prioritize slow clock changes rather than aggressively tracking the remote time source. This helps in situations where the remote time source is known to be poor quality and shows erratic behavior. This can be useful when a stable local clock that smoothly and slowly adjusts to a remote time source is desired. One example would be using a WAN time source (NTP server on the internet perhaps) and serving NTP/PTP to a number of local systems. In this case TimeKeeper would, over time, adjust to track the remote time but would not over-react to timing errors and noise in the source but instead provide a stable clock to the local NTP/PTP clients.

This setting reduces the slew rate that TimeKeeper uses, and time corrections will take much longer than described in the section and graphs above.

Supported platforms

Linux support

TimeKeeper supports all modern x64 Linux distributions (RHEL 6 and newer).

TimeKeeper Compliance is only available on modern x64 Linux distributions (RHEL 6 and newer).

Windows support

TimeKeeper supports the following x64 versions of Windows:

- Windows Server 2022
- Windows Server 2019
- Windows Server 2016
- Windows 11
- Windows 10

Solaris support

TimeKeeper supports the following versions of Solaris (x86 and SPARC architecture only):

- Solaris 10
- Solaris 11

Other platforms and devices

Please inquire about support for other operating systems or specialized devices.

Installation requirements

TimeKeeper requires 150MB of free disk space for the software installation. Additional disk space is required for log files. The space required for log files varies by the number of time sources being tracked and how often.

TimeKeeper currently supports 64-bit x86 and SPARC processors. Please inquire about others if you require support for them. TimeKeeper can be run on x86 systems with processor speed as low as 500MHz.

On startup TimeKeeper will configure a number of process parameters such as stack size and thread/process priority. No special capabilities are required as TimeKeeper will only reduce it's priority. TimeKeeper does require the ability to pin processes and threads to specific CPUs as well as adjust its stack size to 64MB. Some environments administratively prohibit this so TimeKeeper will not operate without being allowed to make these calls.

TimeKeeper Compliance

TimeKeeper Compliance requires 'fontconfig' and fonts be installed in order to generate reports. Some minimal Linux installations may not include these packages by default. Additionally TimeKeeper Compliance only supports modern 64-bit Linux versions (for example, CentOS/RHEL 6).

Updating

These are instructions for updating an *existing* installation, not for installing TimeKeeper for the first time (for that, see the section, "Installing TimeKeeper"). You can update TimeKeeper and its license at the same time or update one or the other independently.

Upgrading TimeKeeper

To upgrade (or downgrade) TimeKeeper, run the installer for the new version then restart TimeKeeper. (See the note about downgrading in the "Log rotation" section.) You do not have to stop TimeKeeper or uninstall the old version first. All of your data files are retained. The same license file from before can be used for the new version, so it is not necessary to have a license file in the same directory as the installer.

On Windows, we *strongly* recommended that you exit all other programs before running the installer. In particular, close those which may be accessing TimeKeeper files or directories, including Windows Explorer.

Updating the TimeKeeper license

To update the TimeKeeper license, replace the existing license file, /opt/timekeeper/release64/timekeeper.lic (on Windows, C:\Program Files\timekeeper\release64\timekeeper.lic), with the new license file. To begin using the new license, restart TimeKeeper then verify in the TimeKeeper log or via the web GUI that the new license is valid and being used.

As of version 8.0.11 when a license file is replaced/updated it is no longer necessary to restart TimeKeeper if the expiration date changed. If the features/options have changed in the license file TimeKeeper will alert that a restart will be automatically performed when the previous license expires.

TimeKeeper previously was sold as a client or a server. As of late 2021 this was simplified so all instances can be a client or server based only on configuration. Client only installations will continue to operate as clients but may also be configured as a server if desired. Users updating an earlier client-only license to one that can operate as a client and server (if configured to do so) will have different features/options as described above. In this case as a one time operation, TimeKeeper should be restarted if a client only license is being replaced with a more capable one, or wait until the client-only license expires fully so that TimeKeeper will restart itself.

Grace period

A license expires on a certain date, but TimeKeeper continues to be available and run normally for some time after that. That extra time is called the grace period.

Up to 30 days before the expiration date, TimeKeeper alerts about the upcoming expiration (see the section, "licenseStateTrap"). After expiration but now in the grace period, TimeKeeper continues to run and can be restarted successfully. During this period, which is typically 30 days, TimeKeeper alerts that it's in the grace period. Once the grace period is over, the TimeKeeper service stops and cannot be restarted until a valid license is installed.

RPM support

TimeKeeper comes with several RPM spec files and FSMLabs provides prebuilt RPMs for some platforms.

RPMs can also be built for your platform by using these 2 spec files found in */opt/timekeeper/release64*:

- timekeeper.spec
- timekeeper-config.spec

These build two separate RPMs for your deployment - *timekeeper.spec* builds the main TimeKeeper RPM, excluding configuration files. There will only be one timekeeper RPM installed at any one time.

The *timekeeper-config.spec* build an RPM contains configuration files, such as the site-specific timekeeper.conf file and the TimeKeeper license file. There will also only be one timekeeper-config RPM installed at any one time. Some users prefer to use only the timekeeper RPM and use Puppet or some other configuration management system to manage configuration data - this is fine. FSMLabs does not provide prebuilt versions of the timekeeper-config RPM.

Prebuilt RPMs

FSMLabs also provides prebuilt RPMs of TimeKeeper for various distributions and versions. TimeKeeper version 8.0.30 and later now deliver RPMs based on *rpmbuild* version 4.14.3. This includes SHA256 digests which can be verified as follows:

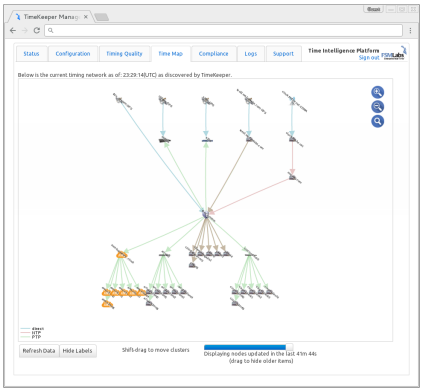
```
# rpm --checksig -v timekeeper-08.00.30-1.x86_64.rpm
timekeeper-08.00.30-1.x86_64.rpm:
    Header SHA256 digest: OK
    Header SHA1 digest: OK
    Payload SHA256 digest: OK
    MD5 digest: OK
```

For more details on getting access to these images, please contact support@fsmlabs.com.

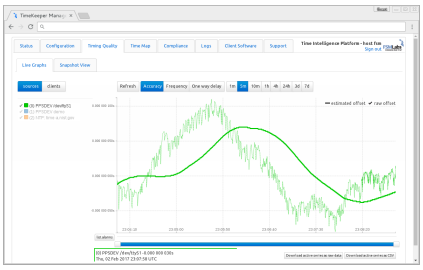
Compliance introduction

Using TimeKeeper, you can quickly verify that your time is in compliance, your timing sources match your intended configuration, and any issues or problems are immediately brought to your attention.

Tools like the timing map quickly let you confirm that clients across your network are receiving time like you intend them to:



Timekeeper of course visualizes your timing sources so you can confirm they're within the expected range. Here we're looking at short term GPS accuracy in the +-50ns range:



TimeKeeper Compliance builds on these core capabilities to provide a centralized reporting tool to track all of your grandmasters, boundary clocks, clients, and other nodes across the enterprise, all from one place.

TimeKeeper Compliance - easy alerting, centralized management

Out of the box, TimeKeeper's easily configured to satisfy common regulatory and compliance requests, such as:

- Report when accuracy requirements are exceeded (and recovered)
- Map the timing network to provide documentation of the timing network
- Log timing accuracy over time, for all sources
- Collect data about clients and peers, roll it all up into a central view

These are common requirements and likely sound familiar. TimeKeeper's built to do this data collection out of the box. TimeKeeper Compliance extends these capabilities to provide simple reports on all of this data, so that you can have daily/weekly/monthly/yearly rollup reports of this information, broken down by host, network, and so on.

Open formats

TimeKeeper maintains all data in clear text formats for easy analysis and management. This includes logging parameters of each configured timing source, client timing quality information, network map data, and so on.

This means the datafiles used in the accuracy plot above is in clear text for archival, or integration with other tools you may need to use. Details on the log formats are in the "File formats" section.

Next we'll look at these tools in Compliance in more detail.

TimeKeeper Compliance overview

Compliance builds on TimeKeeper Server on Linux to provide analysis and reporting for precision time networks for regulatory and data governance.

As events occur on your network, TimeKeeper reports them as needed, whether through syslog, Windows event log, SNMP traps, emails, and so on. As issues occur you get immediate notification however you need.

Compliance extends this capability by reviewing all of your time sources and servers over time and building custom reports based on their behavior. Audits are generated based on custom criteria on a daily, weekly, monthly, and yearly basis. Different groups of systems can be reported on independently, according to different warning and error thresholds based on different reporting requirements.

PDFs are automatically generated for all of these audits for easy reporting and quick display within the Compliance GUI within TimeKeeper. The data used for the reports is stored in a SQL-compliant database that is queryable, allowing for custom reporting if needed. TimeKeeper Compliance tracks a wide variety of clients using PTP, NTP, and other sources to provide timing statistics enterprise wide, all collected and reported from a common platform and interface.

Compliance benefits include:

- Enterprise reporting for precision time networks
- Enhanced analytical tools
- Custom reporting based on different hosts and reporting requirements
- Auditable daily/monthly/weekly/yearly reporting
- SQL interface, Excel export capability, automatic PDF generation
- Data governance

The product is enabled by license and option. To enable it first acquire a Compliance license from FSMLabs, then select the “Enable Compliance” option in TimeKeeper’s Configuration tab.

Your TimeKeeper Compliance package must be configured to receive and respond to management queries and must have active clients or be configured as a PTP client of an active TimeKeeper Server. All clients visible to the TimeKeeper instance will be visible to Compliance. For information on configuring TimeKeeper as a Client or Server, please see the “Configuration” section.

For historical data, historical files must be present. Please allow adequate time for reports to become visible as analysis ends at the end of the most recent hour.

Installation and operating system dependencies

The TimeKeeper Compliance reporting tool is part of TimeKeeper itself. So an installation of TimeKeeper will include the Compliance tool as long as the license enables it. This way no additional software or installations are needed.

Compliance does require some features from the host operating system before it is able to generate reports. Most notably are the font packages used. Those can be installed on most Red Hat/CentOS variants with:

```
yum -y install fontconfig libXrender libXext "*xorg*font*"
```

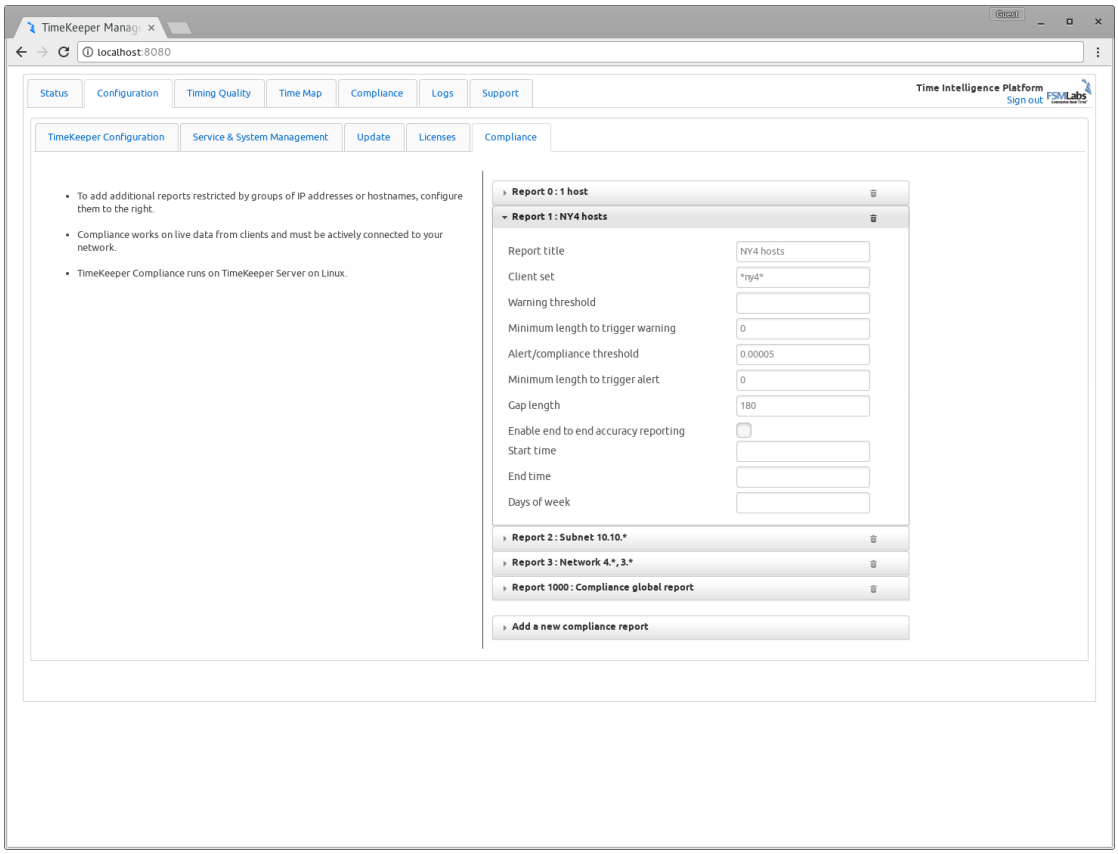
and on Ubuntu systems with:

```
apt-get -y install libxrender
apt-get -y install libxrender1 libfontconfig1
```

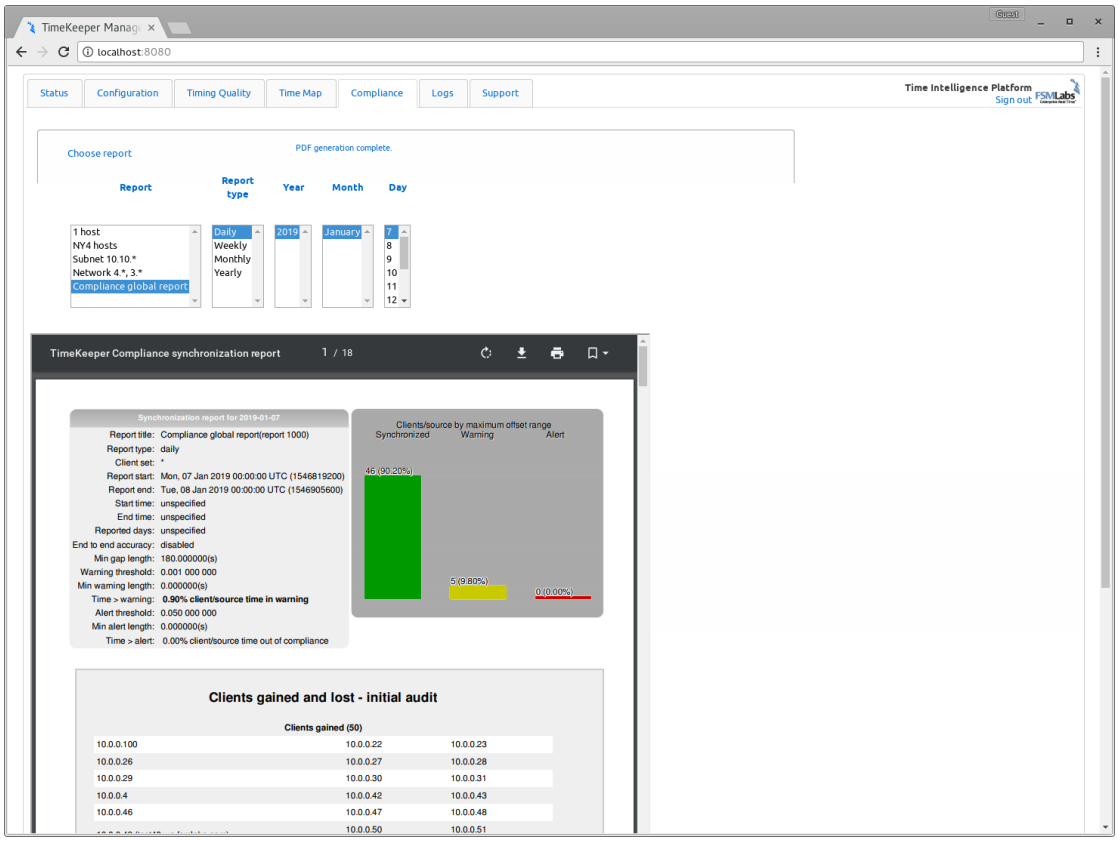
Enabling Compliance

Compliance is enabled or disabled via the GUI by selecting the Configuration tab and selecting the “Enable Compliance” checkbox, on a system with a Compliance-enabled TimeKeeper license. We’ll cover the specifics of this in the next section.

Once enabled, multiple reports can be configured, each for different groups of hosts subject to different alerting and reporting thresholds. Configured via the GUI, reporting configured like this:



Would automatically generate daily audits that look like this:



Looking at the sample in the window above, the audit is a PDF delivered inside the browser, and is also available for download. At the top of the PDF

on the left is an overview of the audit configuration including details and client data that were used at the time that the audit was generated. On the right is a simple histogram indicating which clients were in sync, warning, and alert for the time period of the audit. Note that clients may be reported as being in both warning and alert depending on the events the clients generated.

In the next section, we'll work through the specific configuration steps needed to set up Compliance and use it similar to what's shown above.

Compliance Configuration

Compliance is enabled or disabled via the GUI by selecting the Configuration tab and selecting the “Enable Compliance” checkbox or by adding the following to the TimeKeeper configuration file /etc/timekeeper.conf:

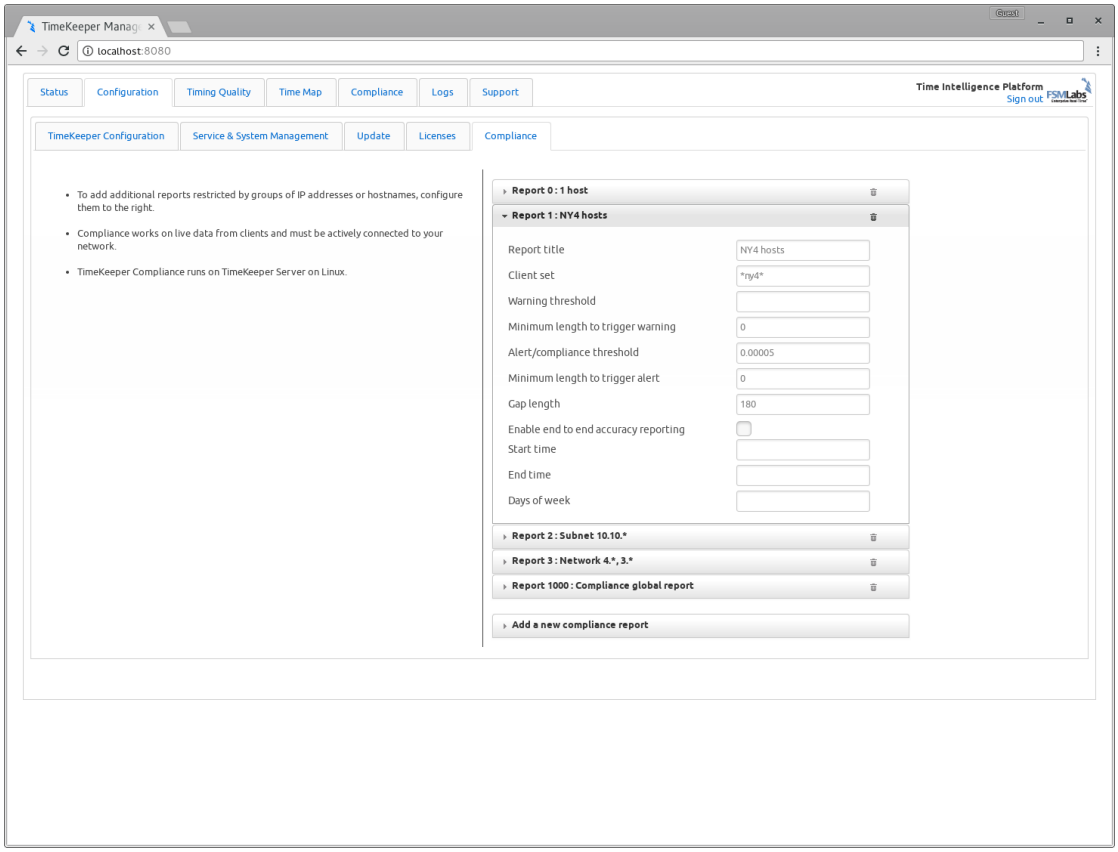
```
ENABLE_COMPLIANCE=1
```

Once Compliance is enabled, reports can be configured in order to provide selective summaries on particular sets of hosts. Instances of those reports are referred to as audits. For example, there is a default report that includes all visible clients on the Compliance instance. This is referred to as the ‘Compliance global report’. Instances of this report for particular day/week/month/year is an audit that provides reporting data on all hosts for that particular time period.

To add custom reports restricted by IP address or hostname you can add an array of COMPLIANCEREPORT() entries in timekeeper.conf as follows:

```
COMPLIANCEREPORT0() { TITLE='1 host'; CLIENTSET='10.78.20.4'; ALERTTHRESHOLD=0.001; }
COMPLIANCEREPORT1() { TITLE='NY4 hosts'; CLIENTSET='*ny4*'; ALERTTHRESHOLD=0.00005; }
COMPLIANCEREPORT2() { TITLE='Subnet 10.10.*'; CLIENTSET='10.10.*'; ALERTTHRESHOLD=0.00005; }
COMPLIANCEREPORT3() { TITLE='Network 4.*, 3.*'; CLIENTSET='4.*,3.*'; ALERTTHRESHOLD=0.00005; }
```

These can be configured in /etc/timekeeper.conf directly or via the web interface. The web interface input would look similar to the following:



Each COMPLIANCEREPORT entry creates an individual PDF for each time period with the ‘CLIENTSET’ and ‘TITLE’ entered. This allows you to create custom reports based on your specific network architecture, hostname groupings, and so on.
The “*” wildcard matches any or no character in the IP address field when used in a CLIENTSET and only those matching the expression are used in each PDF report.

Additionally, multiple criteria in the same COMPLIANCEREPORT’s CLIENTSET can be specified by separating them with commas as in the last example. This way you can group all NY2 and NY4 hosts together for example, while keeping Aurora and London hosts separate. This could look like:

```
COMPLIANCEREPORT0() { TITLE='NY hosts'; CLIENTSET='*-ny4.domain.com,*-ny2.domain.com'; }
COMPLIANCEREPORT1() { TITLE='Aurora hosts'; CLIENTSET='*-aur.domain.com'; }
COMPLIANCEREPORT2() { TITLE='London hosts'; CLIENTSET='*-ldn.domain.com'; }
```

Excluding hosts from a report can be done using “EXCLUDEDCLIENTSET”, e.g.:

```
COMPLIANCEREPORT0() { TITLE='NY hosts'; CLIENTSET='*-ny4.domain.com,*-ny2.domain.com'; EXCLUDEDCLIENTSET='www.*'; }
COMPLIANCEREPORT1() { TITLE='Aurora hosts'; CLIENTSET='*-aur.domain.com'; EXCLUDEDCLIENTSET='www.*'; }
```

This would omit “www.west-aur.domain.com” but include “timeserver.west-aur.domain.com”.

If hostnames/IPs need to be broken down by function rather than geographic region that can be configured for also.

DNS resolution of hostnames for the audits can be enabled or disabled using the configuration key

```
ENABLE_COMPLIANCE_DNS='1';
```

turning off ENABLE_COMPLIANCE_DNS by setting it to 0 results in improved performance but hostnames are not resolved during compliance reporting.

Because different regulations have different reporting thresholds, Compliance allows for different warning and error thresholds. The ALERTTHRESHOLD is an option defined per report so that any client in the named CLIENTSET will be flagged as having an alert threshold breach if the reported accuracy exceeds the named ALERTTHRESHOLD (specified in seconds).

For example, a report may be configured to alert if any ‘trd.domain.com’ hosts report an offset greater than 50 microseconds:

```
COMPLIANCEREPORT0() { TITLE='Trading hosts'; CLIENTSET='*-trd.domain.com'; ALERTTHRESHOLD=0.00005; }
```

Similarly, WARNINGTHRESHOLD allows the configuration of a generally lower threshold that should be watched for as a possible warning of potential problems. For the above case, a warning threshold may be set at 25 microseconds:

```
COMPLIANCEREPORT0() { TITLE='Trading hosts'; CLIENTSET='*-trd.domain.com'; ALERTTHRESHOLD=0.00005; WARNINGTHRESHOLD=0.000025; }
```

This way every daily report will look for clients exceeding a 25us offset and report them as being in a warning state so they can be inspected for issues before exceeding the ALERTTHRESHOLD of 50us. For simplified submitted reports, the warning threshold can be skipped entirely, or configured on a separate ‘internal only’ set of reports.

The ‘Compliance global report’ uses an internal default of 1ms for a warning threshold, and 50ms for an alert threshold. If the global report isn’t needed or wanted in favor of just custom reporting, the global report can be disabled with:

```
ENABLE_COMPLIANCE_GLOBAL_REPORT=0
```

Compliance can also report on end to end accuracy - meaning that in addition to tracking the local accuracy/offset to the client’s time source, it can factor in inaccuracies in getting the time to your local time source. This allows you to build audits based on the reported accuracy values across the entire network.

Configuring a report to generate audits with this feature enabled looks like this:

```
COMPLIANCEREPORT0() { TITLE='Trading hosts'; CLIENTSET='*-trd.domain.com'; ALERTTHRESHOLD=0.00005; ENDTOENDACCURACY=1; }
```

With this enabled hosts will be reported on both reported local accuracy to their source and any inaccuracies reported by the sources themselves. This way audits will reflect the sum of inaccuracies across the network at each point in time.

Compliance reports may also specify the times of the day during which Compliance is to produce reports. These are configured using the parameters STARTTIME and ENDTIME which are specified as offsets from midnight UTC (00:00:00 hours). The maximum duration for a report is 24 hours (86400 seconds).

For instance, New York City weekdays 8-4 time can be represented as follows:

```
COMPLIANCEREPORT0() { TITLE='New York City'; CLIENTSET='*.nyc.your.org'; STARTTIME='13:00:00'; ENDTIME='21:00:00'; DAYS='1,2,3,4,5'; }
```

Tokyo, 8-5 plus daylight savings time, every day except Sunday

```
COMPLIANCEREPORT4() { TITLE='Tokyo'; CLIENTSET='*'; STARTTIME='23:00:00'; ENDTIME='32:00:00'; DAYS='0,1,2,3,4,5'; }
```

For localtimes that change (viz. daylight savings time), using offsets that include both intervals is recommended, starting an hour early for one season and ending an hour late for the other.

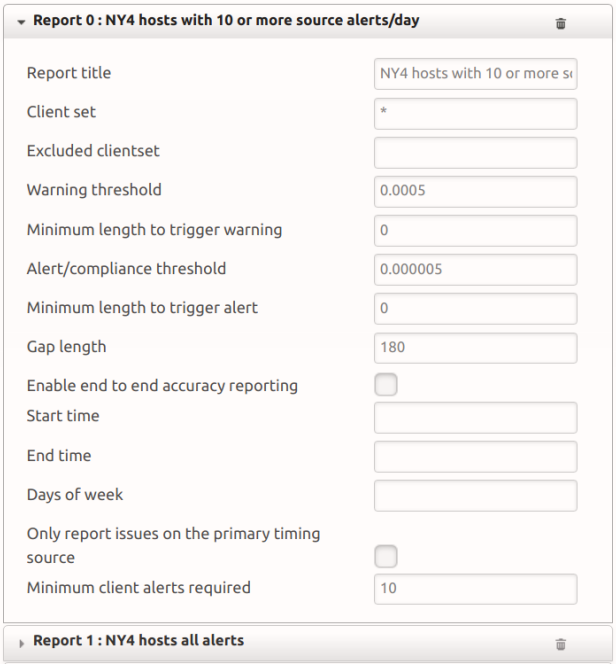
DAYS can be specified either as numbers (0-6 inclusive) or three letter English abbreviations (Sun, Mon, Tue, Wed, Thu, Fri, Sat). The default, blank, is equivalent to "*" which includes all days.

For a full set of Compliance configuration options, please see the rest of the configuration documentation in the "Configuration" section.

Reporting subsets of alerts

Compliance retains a subset of alerts and gaps in data depending on the configuration used. For example, the default gap length is 180 seconds and any client gaps 180 seconds or longer will be summarized in an an audit. A gap of 60 seconds won't be included in the audit with the 180 second threshold set.

Similarly, clients can only provide alerts if there have been enough within a day to require reporting. TimeKeeper can limit alerts with the MINALERTCOUNT configuration option (labeled "Minimum client alerts required" in the web interface) on a per-report basis, as shown below. Please see FINRA CAT requirements for recording issues with persistent errors and thresholds for reporting to see if this feature will be useful for your compliance reporting requirements.



Clients that have less than the configured number of alerts in a configured day (10 in this case) will not be reported on, and their alerts will not be retained or included in summaries. The day considered for alerts by default is 24 hours starting at UTC midnight but may be changed using the STARTTIME and ENDTIME options described above. If a client has enough alerts, they'll be retained in detail and reported on normally.

It can be helpful to also have all of the detail on the individual alerts even if there were fewer than needed to self-report. This is similar to the case where a gap of 180 seconds is reportable and shorter gaps are not recorded but you may also want to also be aware of 60 second gaps internally. An easy way to handle this is to configure additional reports to cover different levels of reporting. One report can be set with tighter restrictions and thresholds for internal tracking, with another set to industry standard thresholds required for external reporting. This way there's a single clear set of data to report with, and another set that can be used to identify issues before they become part of a larger problem.

This type of internal versus external reporting might look like:

```
COMPLIANCEREPORT0() {
 TITLE='External report with minimal alert counting';
 CLIENTSET='';
 STARTTIME='13:00:00';
 ENDTIME='21:00:00';
 DAYS='1,2,3,4,5';
 ALERTTHRESHOLD=0.0001;
 MINALERTCOUNT=10;
}
COMPLIANCEREPORT1() {
 TITLE='Internal reporting with stricter minimal alert count';
 CLIENTSET='*';
 STARTTIME='13:00:00';
 ENDTIME='21:00:00';
 DAYS='1,2,3,4,5';
 ALERTTHRESHOLD=0.0001;
 MINALERTCOUNT=5;
}
```

It's important to note that these alert counts are restricted to alerts that happen within the current day as configured with the optional STARTTIME and ENDTIME options. To report on alert counts that occurred before the start of the configured day, alternate Compliance reports spanning the earlier time window may be configured. Or alternatively, the MINALERTCOUNT value can be set at a lower value for more specific reporting, all the way to a MINALERTCOUNT value of 0 where all alerts are reported. The MINALERTCOUNT=0 value is the default behavior for Compliance reports and audits.

Optimizing Compliance configurations

In many cases, adding the following line to your /etc/timekeeper.conf is helpful:

```
CPU=-1
```

This allows TimeKeeper to run Compliance and the rest of the TimeKeeper suite on all CPUs on the system, which allows for higher throughput and quicker response times.

Also, as Compliance requires a lot of log analysis, disk speed is important. The faster the disk storage is, the faster TimeKeeper Compliance will be able to generate reports on your client data.

In the next section, we'll go over how and where these reports are stored, and how they should be managed.

Compliance tools

In this section we'll go over some general data storage tools and rules that Compliance uses. This is intended to inform users on how to use the product, and also how to make sure proper data retention and archiving processes are applied.

System requirements, CPU, data retention and storage

TimeKeeper Compliance operates by reviewing reported client data. Ideally this data comes from a TimeKeeper client as this has the most complete data, including the client's other source information if it was tracking multiple PTP and NTP sources. Compliance can report on other client data also.

This data must be present on the Compliance instance at the time of audit generation in order to be built into the generated audit. This generally means that client data should be retained on the Compliance server for several days before being archived, in order to handle potential audit regeneration.

Once Compliance has generated the requested audits, the client data can be archived off of the host into long term storage. For some customers, keeping 7 days of data on the Compliance server is sufficient, for others 30 days is preferable. The number of days to keep is configurable in the TimeKeeper logrotate configuration files.

After the data is collected into the configured Compliance audits and taken off of the Compliance server, it can be stored as needed depending on any regulatory requirements. The data has been summarized by Compliance and is no longer needed in raw form unless a particular audit must be regenerated.

Disk space requirements

Disk space required for the raw data (.data files) which Compliance uses to generate reports will vary based on type of source, sync rate and similar. Assuming the default sync rate of 1 update per second and a NTP or PTP time source, each data file (once compressed, which is done automatically) is about 2.7MB per day of data for a single time source. That equates to about 985MB/year. This is about 20TB of data for a year for 20,000 clients. Higher sync rate and more than a single time source per client will increase the disk storage needed.

The reports themselves which are stored in a SQL accessible database (noted below) will vary more. The more often hosts exceed thresholds the more reporting data that must be stored so more disk space will be required.

CPU requirements

When Compliance is processing data it will provide an estimate of how long is required to generate a day of reports and how long until completed. This is shown in the Compliance GUI tab and also in the file:

```
${LOGDIR}/timekeepercompliance/compliance_auditor
```

It is best to test with your own data and hardware (or virtual system), then look at these values to estimate resources required but some estimates of likely performance are below.

Compliance does not require a specific amount of CPU to operate properly. CPU/speed does impact how long it takes to generate reports though. Most users in a steady-state environment will be generating reports every day so incremental reports are run which requires a small amount of processing each day to keep the records/reports updated. When generating reports for a very large amount of data from scratch (creating a new Compliance system that has to process months/years of data) CPU resources become very important.

Predicting how much time is required to generate reports is difficult as it will vary significantly with the number of times clients exceed thresholds (thus more work to do), disk speed and number of CPUs/speed. Assuming a configuration below (which is not intended as a minimal system configuration, simply a reference):

- Configuration includes CPU=-1 (to use all CPUs available)
- System otherwise idle (not dedicating CPU to sharing log files over network with other systems or processing incoming log files)

- 8 Core, 3.6GHz CPU
- 1,100 client systems
- All clients create many dozen threshold violations to report each day
- Rotating (10k) disk

A single day of a single configured report (the default one) on the above system requires about 2.7 minutes/day or 9 hours to process 6 months of data. Normally TimeKeeper will build a report every day so only 2.7 minutes would be required each day. If a user creates a new compliance instance that must process 6 months of data or one deletes the compliance directory and forces TimeKeeper to re-process all data this would require the full 9 hours.

Client data storage

TimeKeeper stores the current client data in this directory:

```
${LOGDIR}/timekeeperclients/
```

The data format of these files is specified in more detail in the "File formats" section. For Compliance to report on a client for a given period of time, that client's data must be present in this directory at the time of audit generation. If it is not included in this directory it will not be included in the generated audit(s).

Once any audits are generated, the data does not need to be kept in this directory and may be stored elsewhere as needed.

Database storage

Compliance stores the data in .db files in this directory:

```
${LOGDIR}/timekeepercompliance/db
```

One file is created per year. By default LOGDIR is set to /var/log, but it may be overridden. No matter where the storage is, FSMLabs recommends that regular backups are made of this content.

PDF storage

In addition to database files, Compliance generates PDF versions of each configured report for every audit period. These files are stored here:

```
${LOGDIR}/timekeepercompliance/pdf
```

with one subdirectory per year below the pdf/ path. It is also recommended that regular backups are made of this directory.

Should PDFs be deleted from this directory, Compliance will regenerate them using the `compliance_cli --rescan` option as detailed below, or the missing PDFs will be flagged for regeneration as of the next TimeKeeper restart. As the original client data was processed earlier and is already present as an audit queryable from the database, the raw client data is not required in order to regenerate PDFs.

The raw client data is not needed to rebuild the PDFs provided the audit was done and is still present in the database.

If there is an issue generating a particular PDF file, Compliance will note it by creating a file named for the intended PDF with a suffix of '.failed'. This is rare but may happen due to resource constraints on an under-resourced system. If this does fail and the issue is remedied (for example by adding memory to the constrained system) the .failed file can be removed. At that point Compliance will reattempt the PDF generation as described above.

Compliance command line tools

compliance_cli

Nearly all Compliance interaction is via the web interface as shown, or by reviewing and distributing the generated PDF audits. However, there a

couple operations that are handled by the `compliance_cli` tool provided in the TimeKeeper installation.

In particular, this tool can be used to trigger the deletion and regeneration of audits. This is generally for situations where the data present at the time of audit generation was incomplete.

Note that deletion of audits is a destructive process and if the client data is no longer present, will result in the loss or modification of regulatory reporting data. This tool should be used with care.

Please contact support@fsmlabs.com with any questions.

Given a particular audit number, type, start time, and year, the `compliance_cli` tool can remove an audit within the Compliance database. The tool can also cause Compliance to rescan the existing client data for this audit period again, regenerating the audit with any corrected data.

compliance_query

A list of existing audits can be retrieved with `compliance_query`. This process may look like:

```
$ /opt/timekeeper/release64/compliance_query --list
(...lists all available audits...)
$ /opt/timekeeper/release64/compliance_cli --year 2017 --start 1509667200 \
    --report 0 --type daily --delete
*** Deleting audits is destructive (deletes data) and may not be recoverable.
*** Please type 'yes' to confirm.
yes
$ /opt/timekeeper/release64/compliance_cli --rescan
```

The last command with *–rescan* informs Compliance that any modifications are now complete and Compliance should scan/reprocess any missing audit periods. The audit(s) will get regenerated but only if client data is present. If the client data is gone or archived elsewhere, the audit may not be regenerated at all or may be incomplete. It is the responsibility of the user to use this tool safely.

Compliance query also allows the user to output data from the compliance database to standard JSON or Excel spreadsheet (xlsx) formats. To use compliance query this way, you can use the following syntax:

```
$ /opt/timekeeper/release64/compliance_query -r 0 -s 1518998400 -y 2018 -t daily \
 -l /app/timekeeperlog/ -f x -o /tmp/my.xlsx
```

Where :

-r is the report number
-s is the start date for the report (`compliance_query --list` provides available values)
-y is the year
-t is the type
-l is the LOGDIR configuration variable value from timekeeper.conf
-f is the output format type (x for xlsx)
-o produces an xlsx file at a location specified at the path given

Running `/opt/timekeeper/release64/compliance_query --help` will list all available command line options.

compliance_generate

Compliance generate allows the user to create reports standalone without needing to enable Compliance in the TimeKeeper configuration file.

Note: Compliance generate needs to be run as root, requires TimeKeeper to be active, and have Compliance disabled in the TimeKeeper configuration.

```
$ /opt/timekeeper/release64/compliance_generate --report 1 -v --title "group1" --clientset "10*" --excludedclientset "*4" \
 --warningthreshold 0.000050 --alertthreshold 0.001  --starttime "08:00:00" --endtime "16:00:00" --days "*"
```

Where :

–report is the report number (this is required)

-v is to enable verbose logging (in the $LOGDIR/timekeepercompliance/compliance_auditor file)

–title is the report title

–clientset is the client list

–excludedclientset is the excluded client set

–warningthreshold is the warning threshold

–alertthreshold is the alert threshold

–starttime is the start time

–endtime is the end time

–days are the days to include in the report

Running `/opt/timekeeper/release64/compliance_generate --help` will list all available command line options. These command line options align directly with the Compliance configuration options available here: Compliance Configuration.

The database files created by *compliance_generate* are located here: *${LOGDIR}/timekeepercompliance/db*. The PDF versions of the generated reports are located here: *${LOGDIR}/timekeepercompliance/pdf*. As of the current release, only the most recent report generated by *compliance_generate* is available in the Compliance GUI. Reports generated earlier are available in the directory: *${LOGDIR}/timekeepercompliance/pdf*.

Compliance database interaction

This section covers the database used by Compliance, tools to query it, and how to use those tools. The examples here are intended to be starting points only, not an exhaustive list of all fields and query methods. The schema is designed to be simple to use out of the box without requiring users to be database experts. The general schema details below will get you connected with the data you need so that you can build the queries you want as quickly as possible.

Note this is for users interested in querying the database directly for their data. This same data is automatically provided in Compliance audits, PDFs, Excel spreadsheet outputs, etc., so users don't have to go to the database directly for data. Direct access to the database is provided and documented here for those users that want to build their own queries and audit summaries.

Compliance database - SQLite

Compliance uses SQLite to store generated audit data. This is a very lightweight and fast database with easy access as a user. There are no other services to run or manage, Compliance writes directly to sqlite .db files in the specified log directory, /var/log by default:

```
${LOGDIR}/timekeepercompliance/db
```

Note that as mentioned previously, backups should be made of this directory for protection in the event of system failures.

Within this directory will be one SQLite database file per year Compliance has stored data for (2018.db for example). In addition, there will be logrotated copies, as many as are configured to be retained in TimeKeeper's logrotate file.

Each .db file is where TimeKeeper Compliance stores results for that year and can be queried directly. Users can open and query that file according to normal Linux permissions. Because these files are self-contained, they can be copied elsewhere if needed, for offline querying or integration.

In many cases command line queries with the tool `sqlite3` is used as we'll see below. However there are many bindings for SQLite and this document isn't meant to cover them all. Users can query the database with whatever tools make sense for their environment. As a precaution copies of the database may be made and queried against elsewhere to minimize interference/risk to the production .db files that are being updated with new data.

Compliance schema

TimeKeeper Compliance uses a contained schema that separates data by individual report/audit. This simplifies management and speeds database interaction. To understand the Compliance schema, it's helpful to keep a couple terms and ideas in mind:

- Reports are what get configured in Compliance to identify a group of hosts with a set of specific thresholds and parameters to meet. (For example, COMPLIANCEREPORT0 in your timekeeper.conf file might identify MiFID II hosts and applicable thresholds in London.)
- Audits are instances of that configured report in your Compliance database. If COMPLIANCEREPORT0 above is a configured report to collect data on hosts in London, the result of what was reported on a specific day/week/etc is an audit.

Compliance stores these audits, each created from a configured report, in individual tables within the database for the year the audit data covers (such as 2018.db in the example above). Each audit table is specific to that report number and to that day/week/month/year.

As we'll see, there is a top level table with details about individual audits. Below that are the individual audit tables with details about the clients in the audit period. Depending on the type of audit there may be additional lower level tables that provide specific additional details about aspects of the audit. We'll start at the top level and work down from there.

Once an audit is complete it's listed in that year's .db file in a top level *audit_names* table. This table is the entry point about a specific audit and period of time. It documents the parameters used at the time of audit generation even if the configured report parameters have changed since. Data includes the following fields and types that may be relevant to users:

- name (VARCHAR) - Name of the table that contains actual audit data
- type (VARCHAR) - Type of audit - daily, weekly, monthly, yearly

- title (VARCHAR) - Title of generating report as it was at time of audit creation
- clientset (VARCHAR) - Clientset of generating report as it was at time of audit creation
- warningthreshold (REAL) - Warning threshold as it was at the time of audit creation
- minwarninglength (REAL) - Minimum warning length as it was at the time of audit creation
- alertthreshold (REAL) - Alert threshold as it was at the time of audit creation
- minalertlength (REAL) - Minimum alert length as it was at the time of audit creation
- mingaplength (REAL) - Minimum gap length as it was at the time of audit creation
- endtoendaccuracy (REAL) - End to end accuracy option as it was at the time of audit creation
- onlyreportprimary (REAL) - Only report primary option as it was at the time of audit creation
- report (INT) - Report number this audit was for
- start (INT) - Start time of audit (in epoch seconds)
- end (INT) - End time of audit (in epoch seconds)

Each audit named in *audit_names* will have its own separate table in the database with details about that particular audit for whatever time period the audit covers. The name of the relevant table will be the name field above. This table has the details about the clients for the audit in question. All audit tables will have the following fields, regardless of type:

- ip (VARCHAR) - IP address of a particular client
- hostname (VARCHAR) - Hostname of that client as of audit generation
- source (INT) - Client's source number reporting the offset data. TimeKeeper clients can provide many different sources
- max (REAL) - Worst offset value in the audit period (for this source)
- max_time (REAL) - Time of worst value in the audit period
- warning_time (REAL) - Time spent in warning in seconds
- alert_time (REAL) - Time spent in alert in seconds
- gap_time (REAL) - Time spent in gap in seconds

Daily audit tables have the additional fields beyond those above:

- start (INT) - Start of data for this day (in epoch seconds)
- end (INT) - End of data for this day (in epoch seconds)
- time_source (VARCHAR) - Source of time at the point of the worst recorded value
- sample_count (INT) - Number of samples for this client during this period

In the case of daily audits, the individual details about specific threshold violations and gaps are also stored in related tables. Alert thresholds are stored in a table of the same name as the daily audit with *_alerts* at the end of the name. Similarly, warning threshold violations are also stored in a table matching the name of the daily audit, but with *_warnings* at the end of the name. Gaps noted in the audit are also stored in a separate table with the same name, with *_gaps* in the name. Details on these tables are below.

Whether a violation and its details are recorded depends on the configured alert and warning thresholds. The threshold that was used in generating this audit is recorded in the *audit_names* table for this audit, in the *warningthreshold* and *alertthreshold* columns. The alerts and warnings violations tables contain the following fields, detailing the specific alert and warning violations (depending on the table) found for each client:

- ip (VARCHAR) - IP address of the client with the issue
- source (INT) - Source number that exceeded the set threshold
- start (REAL) - Start of violation (in epoch seconds)
- end (REAL) - End of violation (in epoch seconds)
- max (REAL) - Worst value of excursion
- max_time (REAL) - Time of worst value above in max
- time_source (VARCHAR) - Source being used for time when the worst value was reached
- uac (REAL) - Upstream/end to end source accuracy at max_time, if known

Gaps may be recorded depending on whether the client's missing data exceeded the gap length in the configured report that generated the audit (default is 180 seconds). The value used will be recorded above for this specific audit in the *mingaplength* field in the *audit_names* entry for this audit. The gaps table contains the following fields:

- ip (VARCHAR) - IP address of the client with the gap
- source (INT) - Source number that experienced the gap
- start (REAL) - Start of gap (in epoch seconds)
- end (REAL) - End of gap (in epoch seconds)
- time_source (VARCHAR)- Source being used for time at the start of the gap

Sample queries

The detail above shows the data structure to enable querying but an example generally makes things more clear. Let's go through the process of getting details on a specific client in a particular day's audit from a report for a specific day, from configuration to individual threshold violation data. In this case let's look for alert threshold violation data for 10.13.153.5 (hostname ukslomfd2trd13) related to configured report 0 on June 11 2018.

First let's start with the configured report in */etc/timekeeper.conf* that will generate the audit. Using a pretty minimal set of parameters the report might look like this:

```
COMPLIANCEREPORT0 () { TITLE='Demo report'; CLIENTSET='ukslo*'; WARNINGTHRESHOLD='.000020'; ALERTTHRESHOLD='.001'; }
```

This will trigger Compliance to build daily audits for any host matching the name 'ukslo*', and record detail like gaps exceeding the default gap length, alerts when clients exceed 1ms, and also report warnings when the clients exceed 20us. With this configured report in place, Compliance will generate audits that can be listed with the *compliance_query* tool, shown here with shortened output:

```
/opt/timekeeper/release64/compliance_query --list
...
2018    daily       0  1528761600  1528848000  (Tue, 12 Jun 2018 00:00:00 - Wed, 13 Jun 2018...
2018    daily       0  1528675200  1528761600  (Mon, 11 Jun 2018 00:00:00 - Tue, 12 Jun 2018...
2018    daily       0  1528588800  1528675200  (Sun, 10 Jun 2018 00:00:00 - Mon, 11 Jun 2018...
2018   weekly       0  1527984000  1528588800  (Sun, 03 Jun 2018 00:00:00 - Sun, 10 Jun 2018...
...
```

Let's get data on the daily audit for report 0 starting on Monday June 11 2018 by querying the top level *audit_names* table described above. `sqlite3` can give you any of the fields about this audit's configuration and summary:

```
$ sqlite3 /var/log/timekeepercompliance/db/2018.db
sqlite> .headers on
sqlite> select name, type, clientset, warningthreshold, alertthreshold from audit_names where start = 1528675200 and report = 0 and type = 'daily';
name|type|clientset|warningthreshold|alertthreshold
daily_1528675200_0|daily|ukslo*|2.0e-05|0.001
sqlite>
```

Again, you can use other tools to query the SQLite database, `sqlite3` is used here as an example. Likewise, you can query the *audit_names* table directly to get a list of audits and all their parameters at once and use that to create later queries, we're just starting with `compliance_query`'s results as an example.

From the above we can see some of the summary data including the thresholds used in creating the audit (20us warning, 1ms alert), the clientset ('ukslo*'), and so on. The name field is the name of the table containing the individual client results for this audit. We can now query that for specifics about the hosts flagged in the audit:

```
sqlite> select ip, hostname, source, max, max_time, alert_time, warning_time from daily_1528675200_0;
ip|hostname|source|max|max_time|warning_time
10.13.153.5|ukslomfd2trd13|0|0.001344818|1528718579.4396|8033.743599914|8033.743599914
10.13.154.23|ukslomfd2trd17|2|0.001115007|1528682525.7795|8010.124199621|8010.124199621
10.12.17.6|ukslomfg2trd77|1|0.001296703|1528704209.9894|8202.531400248|8202.531400248
...
sqlite>
```

The list of hosts in this audit is truncated but 3 entries are shown with their IP address, hostname, the source that had the issue, the maximum offset recorded, the time of that offset, and the amount of time spent in alert/warning for this day.

This is still a high level view - it shows which clients had issues, for how long and what their worst case value was. Other data is available if needed like the source for time when the issue occurred and when exactly the worst case offset happened. You may need to get details about individual

violations beyond the alert threshold, like when they started and ended for a specific host. To get that, we go further to the alerts table for this day, which is named the same as the daily table but with *_alerts* at the end.

To get details about the alerts for 10.13.153.5, that query would look like:

```
sqlite> select start, end, max, max_time, time_source from daily_1528675200_0_alerts where ip = '10.13.153.5';
start|end|max|max_time|time_source
1528682400.9916|1528682750.2848|0.001344647|1528682677.419| NTP(10.13.18.4:10.13.10.66)
1528678800.4958|1528679149.789|0.001343727|1528679144.006| NTP(10.13.18.4:10.13.10.66)
1528686001.4874|1528686350.7806|0.001338525|1528686131.0266| NTP(10.13.18.4:10.13.10.66)
1528693202.479|1528693551.7722|0.00134383|1528693310.0428| NTP(10.13.18.4:10.13.10.66)
1528689601.9832|1528689951.2764|0.001343271|1528689916.5784| NTP(10.13.18.4:10.13.10.66)
1528696802.9748|1528697152.268|0.001344696|1528696815.6974| NTP(10.13.18.4:10.13.10.66)
...
sqlite>
```

This list is truncated but we can see when each alert occurred in time, from the start to the end, what the worst case offset was, when that occurred, and what time source it was using at that point. This can be helpful not only for providing clear reports, but also for identifying common timing issues. Here all of the alerts were triggered when the client was getting NTP from 10.13.18.4, and that may indicate an issue with that NTP server. TimeKeeper's timing map, covered in more detail in the "Timing map and network view" section, can provide insight about that source and identify possible issues.

Likewise, the client's warnings and gaps are in similarly named tables based off of the daily table we started with above.

Again, this interface is provided for those who want to access the data directly to build their own queries and collect their own data. For many users the automatically generated audits available via the web interface, PDF, and Excel output are sufficient but direct SQL access is also provided.

Managing data in the database

Direct modification of the database is not supported and will affect Compliance's ability to provide valid audit data. If audit data is incorrect/incomplete and a specific audit should be rebuilt, please use the *compliance_cli* tool documented in the section, "compliance_cli," in order to regenerate that audit.

Archiving older audit data

Compliance retains all databases and PDFs indefinitely, leaving archiving policy up to the user to define. If you would like to archive older PDFs and databases to an alternate location for long term storage or backup, the process is simple:

- Select the year(s) of database files to archive
- For those same years, archive the same set of PDF files
- Use compliance_cli to rescan for databases, or restart TimeKeeper/Compliance entirely.

For example, if your policy requires archiving 2018 data to offline storage in the path /remote/archive this process would look something like (as root):

```
# cd $LOGDIR/timekeepercompliance
# mkdir -p /remote/archive/compliance/2018/db
# mv db/2018* /remote/archive/compliance/2018/db/
# mkdir -p /remote/archive/compliance/2018/pdf
# mv pdf/2018/ /remote/archive/compliance/2018/pdf/
# /opt/timekeeper/release64/compliance_cli --rescan
```

Once this is complete Compliance will not provide audits with data from 2018 either directly from the database or via the PDF reports. Later if these are required to be served from Compliance directly the process can be reversed.

If you have any questions about how to manage or query the Compliance data you have, please contact support@fsmlabs.com.

MiFID II compliance

The rest of the Compliance documentation covers the details and tools involved in generating compliance reports. However, it may be helpful to demonstrate directly how to make Compliance provide MiFID II reporting for your hosts.

Assume there are a set of hosts subject to the 100 microsecond reporting threshold. For simplicity, let's assume those hosts all have 'accuratehost' in their name - although it could be any set of host names or IP addresses. Hosts may be scattered through the enterprise and need to be separately listed - that would be fine also.

Similarly, there's another group of hosts subject to the 1 millisecond threshold, all on the 10.50.3.x and 10.51.82.x subnets. To generate compliance reports on both of these sets of machines automatically, your Compliance configuration section in /etc/timekeeper.conf would look like this (configured directly or via the web GUI):

```
COMPLIANCEREPORT0() { TITLE='100us hosts'; CLIENTSET='*accuratehost*'; ALERTTHRESHOLD=0.0001; }
COMPLIANCEREPORT1() { TITLE='1ms hosts'; CLIENTSET='10.50.3.*,10.51.82.*'; ALERTTHRESHOLD=0.001; }
```

That's all that's needed. As your Compliance instance collects management data from these hosts it will analyze and build reports on each group separately. New reports will be generated daily, viewable via the web interface interactively, and via PDF files retrievable via the web or on disk. These daily summaries are also collected up into weekly, monthly and yearly reports, all viewable the same way at the same location.

If desired, WARNINGTHRESHOLD values can also be applied so that you get warnings about hosts that are marginal before they exceed the alert threshold. You can do this in the same report above, or create a separate 'internal' report that includes the warning threshold for simpler external reporting.

FINRA CAT compliance

TimeKeeper's multisource feature, the license-controlled ability to use more than one source among other things, allows easy compliance with FINRA CAT NMS Plan requirements. In general for Compliance configuration, the MiFID II and Compliance guidelines apply similarly to systems bound to FINRA CAT thresholds.

Accuracy to within 50 milliseconds of NIST time is required - but it's unlikely that a public NIST server is being used to drive the clock. By pointing to a NIST server as a secondary source, accurate logs can be maintained that provide continuous system time offsets from NIST time.

To configure for this scenario, the TimeKeeper configuration should specify a NIST server as a secondary (or even lower priority) time source, as in:

```
SOURCE0() { PTPCLIENTVERSION=2; PTPDOMAIN=80; }
...
SOURCE10() { NTPSERVER=time-a.nist.gov; NTPSYNCRATE=0.1; }
```

In this case, the primary source is a local PTP Grandmaster on domain 80. The NIST server time-a.nist.gov is a non-primary source that will be tracked and logged in *timekeeper_10.data* in */var/log* by default. Additional servers can be specified in order to maintain a tracked offset if source 0 goes offline.

/var/log/timekeeper_10.data will look approximately like this:

```
1486063703.142371588 -0.000033029 0.000013451 ...
1486063709.244078475 -0.000027511 -0.000001339 ...
1486063726.824376694 -0.000022426 0.000111881 ...
1486063738.072544688 -0.000009811 0.000101470 ...
```

As described in the "File formats" section, the first column is the time reported by the time source (time-a.nist.gov in this case). The second column is the local time offset from that sample. These public NTP servers are subject to variations in round trip time across the public internet, so they will not report a sync as clean as high quality local time sources like we configured for source 0 (/var/log/timekeeper_0.data). TimeKeeper applies the same

smoothing algorithms it would to any time source, though.

TimeKeeper constantly steers the clock based on the primary source at every sample, so there is no need to explicitly synchronize the servers by hand at predetermined intervals.

When using the multisource feature, TimeKeeper will fall back to other sources (1-9) in the above configuration if the primary PTP grandmaster ceases to provide PTP data. This means there is an automatic fallback in place, even for a different protocol. If all of the other sources fail TimeKeeper will steer to the NIST source, although in general given the long links and noisy network involved the accuracy would be greatly reduced compared to an accurate local failover source.

/var/log/timekeeper_10.data files maintain a complete history of the local clock offset from NIST time and should be retained with other audit trail data. When referencing a public NTP server, please comply with that server's usage guidelines.

TimeKeeper recipes

TimeKeeper can be used as a client, server, grandmaster, compliance tool, and more. Other sections of this documentation cover specifics of features related to those scenarios. Here the focus is on introducing TimeKeeper in narrow configurations, starting as a client, in order to concentrate on core principles.

In order to focus on the core idea of each recipe, extra options that would be seen in a typical deployment are generally omitted. For example, when failing over between sources an SNMP trap host might be configured to send alerts about the change. A recipe about failover will omit those SNMP configuration details to focus on the core feature at hand, which is failover.

Please refer to either other recipes or other sections of the documentation for broader coverage of all of the options TimeKeeper offers.

Client recipes

1. Client with PTP and NTP failover
2. Alert and send SNMP traps when offset exceeds threshold

Using TimeKeeper with Telegraf/InfluxDB/Grafana (TIG stack)

The Telegraf/InfluxDB/Grafana (TIG) stack is a set of open source tools to collect, store and graph time series data. This document describes how to install and set up the TIG-stack for use with TimeKeeper. Collecting of the data can be done by Telegraf or directly by scripting. InfluxDB is used to store the data. Finally the data can be graphed or viewed using Grafana.

TimeKeeper has three types of log files:

1. timekeeper (timekeeper.log on Windows) - central place for logging alarms, general messages, state change information and administrative information
2. timekeeper_$SOURCE.data - one file per configured source
3. timekeeperclients/ - directory of per-host and per-source data files for each reporting client

More information on TimeKeeper's log files can be found in the "File formats" section.

InfluxDB

InfluxDB is an open source time series database. TimeKeeper provides excellent integration with InfluxDB. In this section we'll explain one possible integration. The InfluxDB reference manual is available at https://docs.influxdata.com. These examples and documentation apply to version 1.8 of InfluxDB and you may need to adjust depending on the distribution in use.

The influx API is available on port 8086 by default. Here is an example query:

```
curl -G http:/127.0.0.1:8086/query?pretty=true --data-urlencode "q=show databases"
```

Below is an example bash script to process a specific timekeeper data file and import it into InfluxDB.

```
#!/bin/bash
DB=timekeeper_db
DB_SERVER="http://127.0.0.1:8086"
CURL_OPTS="-w %{http_code} -s -o /dev/null"
FILE=${1}
if [ ! "$FILE" ] ; then
 echo Missing file name. Must be something like timekeeper_0.data.
 echo Usage: $0 FILENAME
 exit 1
fi
SOURCE=`basename ${FILE} | sed 's:timekeeper_::g' | sed 's:\.data::g`
#
# Make sure we can access the InfluxDB
#
STATUS=`curl ${CURL_OPTS} -G  ${DB_SERVER}/query?pretty=true --data-urlencode "q=show databases"`
if [ $? != 0 ] ; then
 echo "Unable to access Database"
 exit 1
fi
#
# Drop timekeeper_db
#
STATUS=`curl ${CURL_OPTS} -i -XPOST ${DB_SERVER}/query --data-urlencode "q=drop database ${DB}"`
if [[ ${STATUS} != 200 ]] ; then
 echo "Status (${STATUS}) Not expected dropping database ${DB}"
 exit 1
fi
#
# Create timekeeper_db
#
STATUS=`curl  ${CURL_OPTS} -i -XPOST ${DB_SERVER}/query --data-urlencode "q=create database ${DB}"`
if [[ ${STATUS} != 200 ]] ; then
 echo "Status (${STATUS}) Not expected creating database ${DB}"
 exit 1
fi
#
# Write timekeeper data to db
#
while IFS= read LINE; do
 TIMESTAMP_1=`echo $LINE | awk '{ print $1 }' | sed 's:\.::g`
 OFFSET_2=`echo $LINE | awk '{ print $2 }'`
 RAW_OFFSET_3=`echo $LINE | awk '{ print $3 }'`
 INTERNAL_4=`echo $LINE | awk '{ print $4 }'`
 CCF_5=`echo $LINE | awk '{ print $5 }'`
 ADD_DATA_GPS_6=`echo $LINE | awk '{ print $6 }'`
 ADD_DATA_GPS_7=`echo $LINE | awk '{ print $7 }'`
 INTERNAL_8=`echo $LINE | awk '{ print $8 }'`
 TS_TYPE_9=`echo $LINE | awk '{ print $9 }'`
 NET_DELAY_10=`echo $LINE | awk '{ print $10 }'`
 RAW_NET_DELAY_11=`echo $LINE | awk '{ print $11 }'`
 TIME_SRC_12=`echo $LINE | awk '{ print $12 }'`
 IDEAL_CCR_13=`echo $LINE | awk '{ print $13 }'`
 MODEL_ERROR_14=`echo $LINE | awk '{ print $14 }'`
 SRC_TRC_15=`echo $LINE | awk '{ print $15 }'`
 INTERNAL_16=`echo $LINE | awk '{ print $16 }'`
 UPSTREAM_ACC_17=`echo $LINE | awk '{ print $17 }'`
 INTERNAL_18=`echo $LINE | awk '{ print $18 }'`
 STATUS=`curl  ${CURL_OPTS} -i -XPOST "${DB_SERVER}/write?db=${DB}" --data-binary "tk_data,source=${SOURCE} offset_2=${OFFSET_2},raw_offset_3=${RAW_OFFSET_3},internal_4=${INTE
add_data_gps_7=${ADD_DATA_GPS_7},internal_8=${INTERNAL_8},ts_type_9=\"${TS_TYPE_9}\",net_delay_10=${NET_DELAY_10},raw_net_delay_11=${RAW_NET_DELAY_11},time_src_12=\"${TII
ideal_ccr_13=${IDEAL_CCR_13},model_error_14=${MODEL_ERROR_14},src_trc_15=\"${SRC_TRC_15}\",internal_16=${INTERNAL_16} ${TIMESTAMP_1}"`
 if [[ ${STATUS} != 204 ]] ; then
   echo "Status (${STATUS}) Not expected adding record to database: ${DB}"
   exit 1
 fi
done < ${FILE}
```

Telegraf

Telegraf is an agent for collecting, processing, aggregating and writing metrics. Telegraf's documentation can be found here:

https://docs.influxdata.com/telegraf/. These examples apply to version 1.18 of Telegraf and can be adapted to your version as appropriate.

The Telegraf configuration file is located at: */etc/telegraf/telegraf.conf*. To enable Telegraf to consume TimeKeeper logs you can use either the *inputs.file* or the *inputs.tail* plugins. Examples are available below.

> Note: Only one type of plugin, either *inputs.file* or *inputs.tail*, should be used with the TimeKeeper logs.

The *inputs.file* section in telegraf.conf parses the complete contents of the file with the selected format. To import existing TimeKeeper data using Telegraf add the following to telegraf.conf:

```
[[inputs.file]]
  files = ["/var/log/timekeeper_0.data"]
  data_format = "csv"
  csv_timestamp_column = "ts_1"
  csv_timestamp_format = "unix"
  name_override = "tk_data"
  csv_header_row_count = 0
  csv_column_names = ["ts_1","offset_2","raw_offset_3","internal_4","ccf_5","add_data_gps_6","add_data_gps_7","internal_8","ts_type_9","net_delay_10","raw_net_delay_11","time_source_12","ideal
  csv_column_types = ["float","float","float","float","float","float","float","float","string","float","float","string","float","float","string","float","float","float","float"]
  csv_delimiter = " "
  [inputs.file.tags]
    source = "0"
```

The *inputs.tail* section in telegraf.conf “tails” a file and parses each update. To continuously import the the TimeKeeper log file add the following to telegraf.conf:

```
[[inputs.tail]]
  files = ["/var/log/timekeeper"]
  from_beginning = true
  grok_patterns = ["%{NUMBER:ts:float}: %{GREEDYDATA:log}"]
  fielddrop = ["ts"]
  name_override = "tk_log"
  grok_custom_pattern_files = []
  grok_custom_patterns = '''
  '''
  data_format= "grok"
```

To continuously import the the TimeKeeper data file using the *inputs.tail* plugin add the following to telegraf.conf:

```
[[inputs.tail]]
files = ["/var/log/timekeeper_0.data"]
from_beginning = true
grok_patterns = ["%{NUMBER:ts_1:float} %{NUMBER:offset_2:float} %{NUMBER:raw_offset_3:float} %{NUMBER:internal_4:float} %{NUMBER:ccf_5:float} %{NUMBER:add_data_gps_6:float} %{NUM
fielddrop = ["internal_4", "internal_8", "internal_16", "internal_18", "internal_19"]
name_override = "tk_data"
grok_custom_pattern_files = []
grok_custom_patterns = '''
'''
data_format= "grok"
[inputs.tail.tags]
  source = "0"
```

To configure Telegraf to update a file use:

```
[[outputs.file]]
files = ["/tmp/telegraf.out"]
data_format= "influx"
```

To configure Telegraf to update and existing InfluxDB use:

```
[[outputs.influxdb]]
  urls = ["http://127.0.0.1:8086"]
  database = "timekeeper_db"
```

> Note: The precise TimeKeeper timestamp may not be retained when importing data into InfluxDB using Telegraf. As configured this is intended to be used for visualization and analysis and not for regulatory reporting.

The telegraf utility is a command line tool to access Telegraf. A useful command for testing an updated telegraf.conf file is:

```
telegraf --debug --config telegraf.conf  #(Use to validate and debug a Telegraf configuration file)
```

Grafana

Grafana is an open source data analytics and visualization web application. Grafana documentation can be found here: https://grafana.com/docs/. The examples and instructions provided below apply to version 7.5.2 but can be adopted to whatever version you're using as needed.

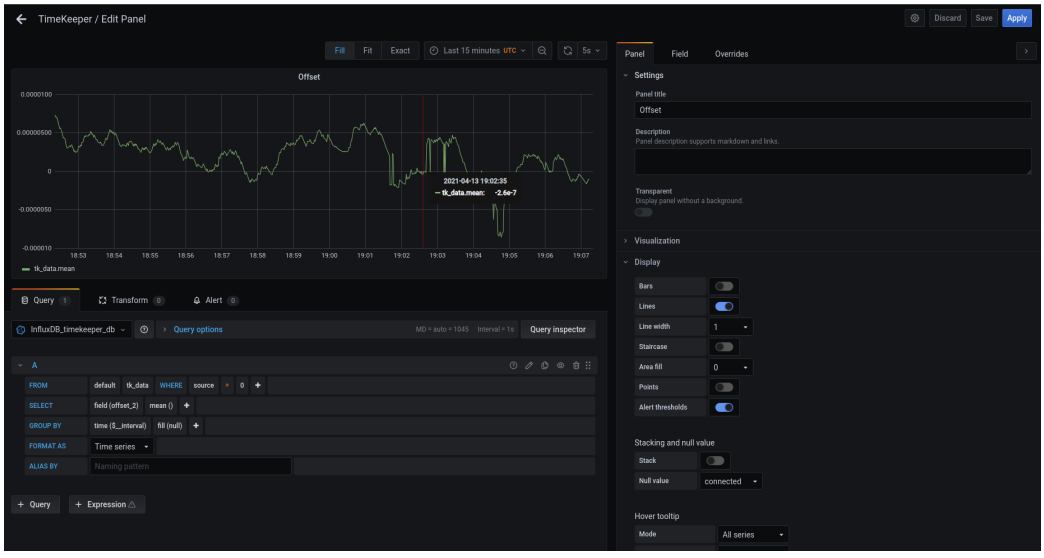
Viewing TimeKeeper Logs and Data in Grafana

Enable the InfluxDB TimeKeeper data source

1. In Configuration/Data Sources select Add data source and InfluxDB.
2. Set the URL to be: http://localhost:8086
3. Select the database to be timekeeper_db (Or the database used to import the TimeKeeper data)
4. Use Save & Test to validate the Data source.

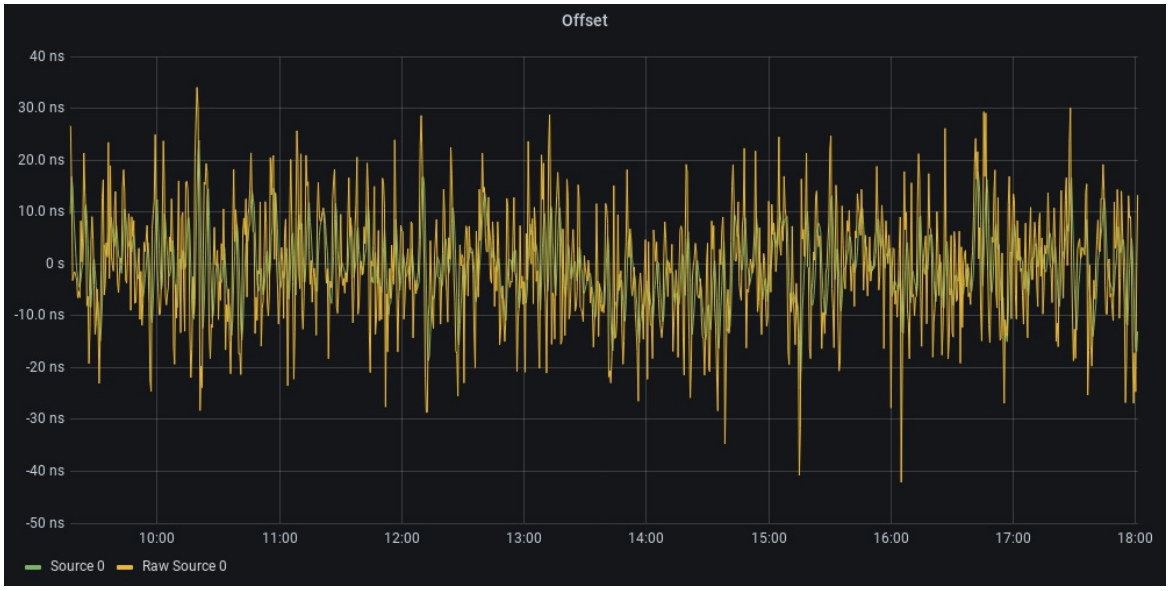
Create Dashboard and Panel to view TimeKeeper Data

1. Create Dashboard
2. Create Panel for Offset
    1. Use Visualization *Graph*
    2. For the Panel Title use *Offset* (In the configuration sidebar Panel under Settings)
    3. Set the "Area fill" *0* (In the configuration sidebar Panel under Display)
    4. Set the "Null Value" to *Connected* (In the configuration sidebar Panel under Display, see screenshot below). *Note* that these graph options may change in future versions of Grafana

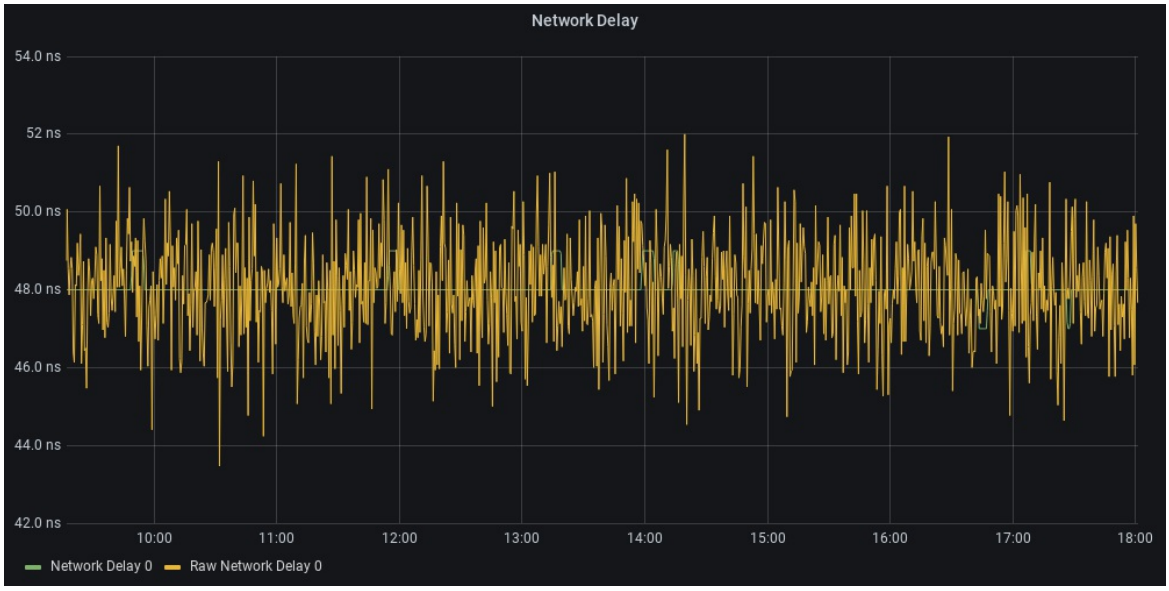


    5. Axes/Left Y Unit *Misc/Time/Seconds*
    6. Ensure that the data source selected is the one created above
    7. For Accuracy(estimated offset) click on the pencil to *toggle text edit mode* and enter: `SELECT mean("offset_2") FROM "tk_data" WHERE ("source" = '0') AND $timeFilter GROUP BY time($__interval) fill(null)`
    8. Set the *ALIAS BY* to Source 0
    9. Use the "+ Query" button to create and additional query for the raw offset

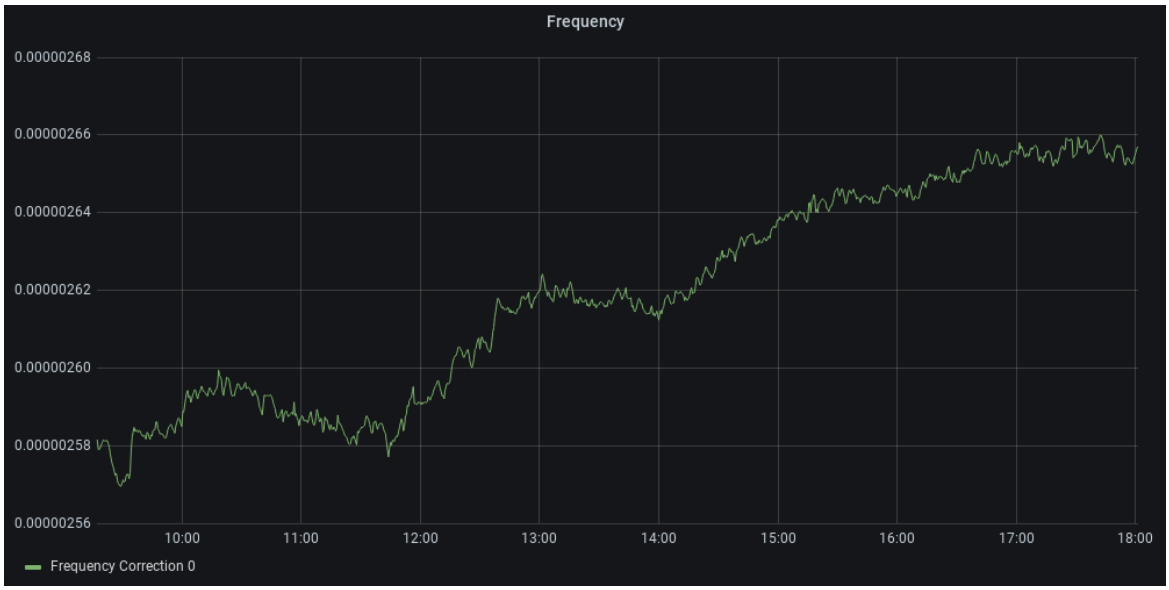
10. For Accuracy(raw offset) click on the pencil to *toggle text edit mode* and enter: `SELECT mean("raw_offset_3") FROM "tk_data" WHERE ("source" = '0') AND $timeFilter GROUP BY time($__interval) fill(null)`
11. Set the *ALIAS BY* to Raw Source 0
12. See example *Offset* panel below



3. Create Panel for Network Delay/One way delay
    1. Use the settings above for the Panel
    2. For the Panel title use *Network Delay* of *One way delay* (This is available in the configuration sidebar under Settings)
    3. Set the “Area fill” *0* (In the configuration sidebar Panel under Display)
    4. Set the “Null Value” to *Connected* (In the configuration sidebar Panel under Display)
    5. For the calculated Network Delay/One way Delay use *toggle text mode* and enter: `SELECT mean("net_delay_10") FROM "tk_data" WHERE ("source" = '0') AND $timeFilter GROUP BY time($__interval) fill(null)`
    6. Set the *ALIAS BY* to Network Delay 0
    7. Use the “+ Query” button to create and additional query for the raw network delay
    8. For the raw Network Delay/One way Delay use *toggle text mode* and enter: `SELECT mean("raw_net_delay_11") FROM "tk_data" WHERE ("source" = '0') AND $timeFilter GROUP BY time($__interval) fill(null)`
    9. Set the *ALIAS BY* to Raw Network Delay 0
    10. See example *Network Delay* panel below



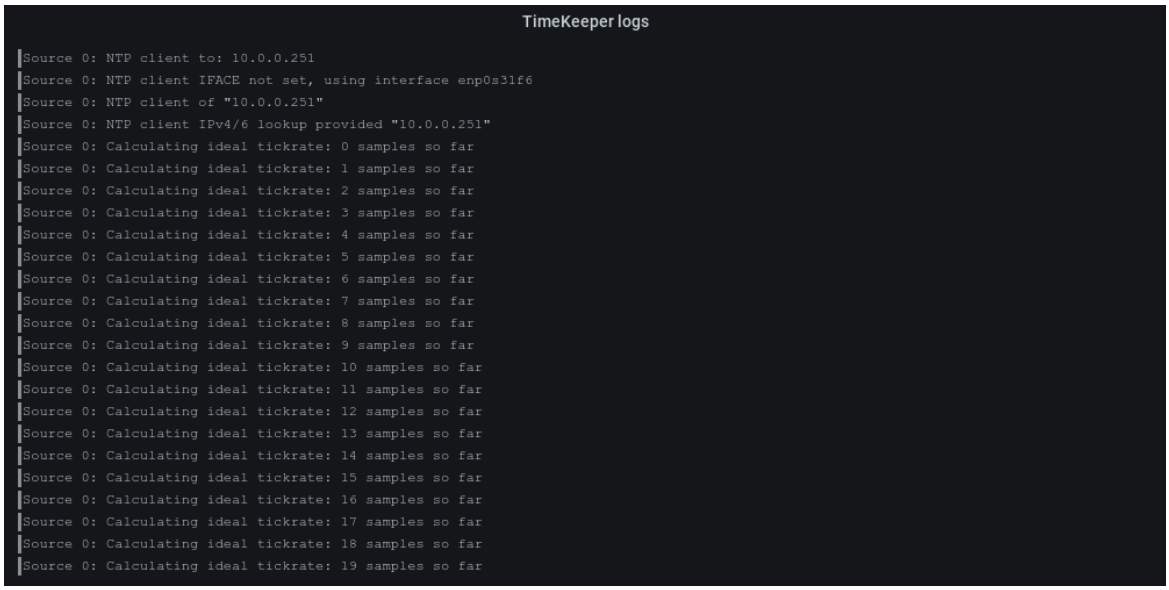
4. Create Panel for Frequency
    1. Use Visualization *Graph*
    2. For the Panel title use *Frequency* (This is available in the configuration sidebar under Settings)
    3. Set the “Area fill” *0* (In the configuration sidebar Panel under Display)
    4. Set the “Null Value” to *Connected* (In the configuration sidebar Panel under Display)
    5. Set the *ALIAS BY* to Frequency Correction
    6. For the calculated Frequency Correction use *toggle text mode* and enter: `SELECT mean("ideal_ccr_13") FROM "tk_data" WHERE ("source" = '0') AND $timeFilter GROUP BY time($__interval) fill(null)`
    7. Set the *ALIAS BY* to Frequency Correction 0
    8. See example *Frequency* panel below



View TimeKeeper Logs

1. Create Panel for TimeKeeper logs
    1. Use Visualization *Logs*
    2. Select the data source to be timekeeper_db
    3. Select Measurement to be tk_log
    4. Select field to be “field(log)”
    5. Change mean() to Aggregations/distinct()
    6. Set “FORMAT AS” to “Table”

    Note: Steps 3-6 can be replaced by clicking on the pencil to toggle text edit mode and enter: `SELECT distinct("log") FROM "tk_log" WHERE $timeFilter GROUP BY time($__interval) fill(null)`



Notes

1. A time selection in the Offset/Network Delay/Frequency will cause the TimeKeeper logs for that time frame to be visible. However a similar selection in the TimeKeeper logs is not available.
2. In the above examples the TimeKeeper logs shows how to enter the InfluxDB query using the UI elements (as opposed to the “text edit mode”). This is easier for developing new panels and allows for easier tweaking of the output. Toggling the text edit mode will cause the resulting query to be displayed.

Advanced topics

This section discusses some more advanced topics that are not part of initial setup and deployment of TimeKeeper. This section is more focused on special cases, configurations and environments that are either uncommon or have special requirements.

Virtual environments

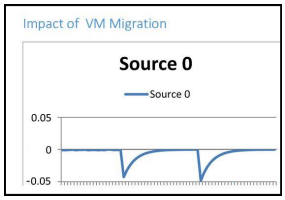
TimeKeeper can handle the unique environment of the cloud and virtual hosts. However, it's important to understand the environment for time sync in virtualized systems in order to configure TimeKeeper and set expectations properly. On virtual hardware the clock is virtualized and may show discontinuous gaps in time which does not happen on physical systems. As a result applications depending on legacy time synchronization technology in the cloud see time jumps, lurches, and divergences which cause misbehavior. TimeKeeper recognizes this environment and is designed to synchronize the clock properly. With care and proper configuration it's possible to have near bare-metal hardware accurate time. However, virtual host operations can impact the accuracy of time on these systems so expectation should be set appropriately and users must be aware of these issues. We describe those below.

Common cloud/virtual issues and solutions

Migration awareness

TimeKeeper detects when a virtual server instance is migrated or stopped then restarted and keeps time consistent for all applications. This means that timestamps, log files and application output are correlated throughout the network as images move around. In general we recommend disabling migration or stopping the virtual instance for long periods of time. That is because the time will immediately show an error of the amount the operation required. For example if it takes 3 seconds to migrate the virtual instance to a new host, TimeKeeper will show a 3 second error. That is because the virtual clock has been stopped for that long. When it wakes up, the time is incorrect–it's behind the source time, so the offset suddenly goes negative.

TimeKeeper will immediately recognize this and adjust the clock rate as quickly as possible to correct the error smoothly. TimeKeeper will not "jump" the time to make corrections unless explicitly configured to do so (see ALLOW_SET_TIME_AFTER_STARTUP in the "Global options" section)'. It smoothly speeds and slows the clock to adjust time via a controlled slew. Traditional and non-VM aware time synchronization tools will allow VM introduced errors to accumulate and then adjust the error all at once, then allow more error to accumulate. Applications are often not tolerant of time jumping, lurching and making abrupt changes. Logs will show incorrect times and multiple servers tracking the same time source will show wildly varying times. TimeKeeper makes time predictable and stops that behavior. This can cause a sawtooth pattern in the time offset graphs when these errors are introduced. The image below shows that.



Pauses in operation of the virtual instance can be many seconds and quite dramatic or for a few milliseconds at a time. The latter is common when physical hardware is overloaded and cannot service all virtual hosts. The above time errors can appear frequently in those cases. It's best to reduce the load on the physical hardware in those cases to allow the virtual machines to run properly without the pauses that introduce time errors.

It is possible to configure TimeKeeper to immediately correct large time errors rather than smoothly adjust the time (which can take a while). Enabling the option below will cause TimeKeeper to correct any large time errors immediately. By default, an error of 5 seconds or more is required for TimeKeeper to make this correction.

```
ALLOW_SET_TIME_AFTER_STARTUP=1
```

TimeKeeper also provides the SET_TIME_THRESHOLD option to specify a different threshold. Here where a 50ms error will be immediately corrected for:

```
SET_TIME_THRESHOLD=0.05
```

Using this option allows VM instances held up by common virtualization overhead (resource contention, migration, etc) to be quickly brought back into sync rather than letting TimeKeeper slowly correct for the error. In situations where periodic delays are unavoidable, this option can make sure clock

accuracy remains within regulatory limits. Because TimeKeeper will validate an offset before allowing the clock to step, this process can take as long as is required to process several PTP/NTP queries. If it's important to step the clock as quickly as possible, it's recommended that clients be configured to query at a higher than normal sampling rate so that those validation steps can be completed faster.

Note that SET_TIME_THRESHOLD can also affect startup behavior. If SET_TIME_ON_STARTUP is not set, TimeKeeper steers the clock (rather than step) to incrementally correct any offset error if the offset is less than the value of SET_TIME_THRESHOLD. This means that setting SET_TIME_THRESHOLD to a different value (from the 5 second default) in order to change the threshold where the clock is stepped post startup due to VM delays also changes the threshold where TimeKeeper will step the clock at startup. Setting SET_TIME_ON_STARTUP means TimeKeeper always sets the clock on startup regardless of offset and SET_TIME_THRESHOLD only affects post startup behavior.

Good time sources

Something often overlooked in cloud and virtual environments but not with physical hardware is the quality of what is providing time to the client. Time servers in most cloud environments, if they are even available, provide very poor accuracy. Users often have very little choice in what to use, too. In some self-hosted virtual environments that may not be the case due to availability of local time servers managed by a local team. In cases where a 3rd party provides time and manages the platforms one must take care in picking where time comes from.

Some cloud providers, such as AWS, provide local time servers but our testing has shown that these may not be the best choice. The accuracy of these sources can vary greatly and be erratic so thorough testing is suggested. It's sometimes best to use internal time servers even when going across the WAN to get to them. See more in the "Timing Architecture" section below.

Cross-check of time

The cloud presents a unique challenge when trying to verify time sync. Some cloud providers do not offer traceability to UTC (required for some regulatory reporting). Doing cross-checks against other time sources often require going over the same network links that the primary time source uses and means the cross-check suffers the same problems the main time source does.

In all setups, but especially with cloud and virtual system, one needs to check that the time one is being fed is accurate. A single bad network segment or bad time server should not cause you to have bad time and TimeKeeper won't allow that to happen if configured properly. TimeKeeper is able to track multiple time sources at the same time and monitor the primary time source for quality and alert when there is a problem and optionally to take corrective action immediately - even in the presence of jamming or spoof attack. This is often a case-by-case issue that is specific to the particular setup and sources available.

Timing architecture in cloud environments

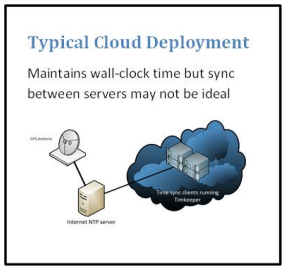
Accurate time synchronization in cloud environments is not difficult to setup but there are some things one must know. New financial regulations (MiFID II and FINRA changes) require 1 millisecond and sometimes 100 microsecond accuracy to UTC time as well as traceability to UTC. Many social media, gaming and other applications require similar accurate timing. In the past that was not easy since these are virtualized environments where system time does not always match "real time". Getting past that is not difficult now though. Below we describe some possible configurations and tradeoffs using a cloud service to show how to use TimeKeeper to achieve those time accuracy milestones.

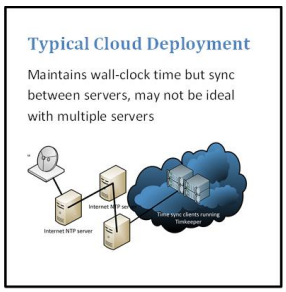
It's not unusual to see 10 minutes/day drift with non-time synchronized AWS EC2 instances, given the harsh environment for time sync in the cloud. Precise time requires the sophisticated algorithms, filters, and network timing models that TimeKeeper provides.

It's also very important to properly setup your clients to get time from a good quality time server. Below you can see how a typical setup normally works. AWS does not provide internal time servers so one has to get time from the internet. Often that means whatever default time servers a particular version of Windows or Linux time sync clients defaults to. That's often a very bad choice. Even with GPS-backed TimeKeeper NTP (equivalent to UTC here) as the original time source over the internet there may 3 intermediate time servers adding their own errors before it reaches the final client system. TimeKeeper automatically maps time sources that allows you to visualize this and check for a similar setups.

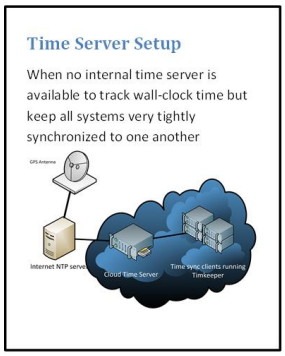
Below are typical cloud configurations. They use external (outside of the hosting cloud) servers which often are themselves getting 2nd or 3rd hand

time from other servers. It's both a typical/default setup and also the worst quality time.



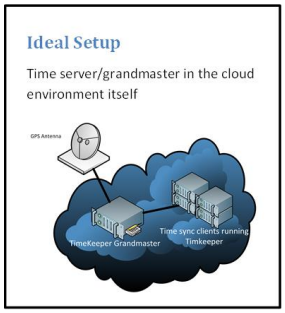


Below is a far better setup. A single cloud instance is used to track a good time server. That in turn provides time to the other systems within the cloud. This will allow close tracking of GPS/UTC on one server that can clean up the time signal (remove jitter, stabilize any WAN noise, etc) while also allowing for a very tight time sync between instances.



This above achieves the most common need in cloud environments. That is to keep a group of servers as tightly synchronized as possible with that time then being reasonably close to UTC (wall-clock time). TimeKeeper can handle these competing requirements by tracking an absolute time source (UTC) and providing that time to peer systems in the cloud. This allows a collection of servers to be managed as a single unit.

This is an ideal setup. This is only available when a GPS time server is available inside of your own network. TimeKeeper is configured to source time directly from that system over a leased line or other direct link from within the cloud.



VM Configuration Guidance

Hypervisor	Virtual Machine Configuration	Guest Integration Configuration
VirtualBox	VBoxManage setextradata VM-name "VBoxInternal/Devices/VMMDev/0/Config/GetHostTimeDisabled" 1	Ensure Guest Additions do not modify the clock by searching Oracle's web site for "VirtualBox Disabling the Guest Additions Time Synchronization"

Hypervisor	Virtual Machine Configuration	Guest Integration Configuration
VMware	tools.syncTime = FALSE time.synchronize.continue = FALSE time.synchronize.restore = FALSE time.synchronize.resume.disk = FALSE time.synchronize.shrink = FALSE time.synchronize.tools.startup = FALSE time.synchronize.resume.host = FALSE	*N/A*
Linux KVM	*N/A*	Ensure clock-steering features are disabled in chosen toolset (i.e. virt-manager, QEMU guest agent, etc)

PTP aware switches

Some switches (and other network gear) are PTP aware. They participate in the exchange of time using the PTP protocol as either “boundary clocks” or “transparent clocks”. We generally and strongly recommend all these features be disabled.

Our testing and our experience has shown that (except in limited circumstances) these features are often buggy, introduce time errors, connectivity errors and are often very hard to track down. We do not maintain a database or matrix of equipment and firmware versions that do and do not work because that is an ever changing landscape so we are unable to provide recommendations when using these features.

PTP aware switches add a further complication because they will often block PTP monitoring/management traffic. These messages are necessary in order to properly monitor client accuracy/behavior, maintain traceability logs and comply with many financial regulations for time sync.

The best time sync accuracy and reliability will come from disabling all of these features and making switches unaware of PTP entirely.

CPU consumption

TimeKeeper will move itself to the last CPU on the system (highest numbered CPU) unless configured to do otherwise with the CPU= configuration variable. This is designed to reduce the impact on other applications. If a user wants to reduce TimeKeeper impact further by reducing CPU consumption there are a few options available to do that.

Source accuracy

The greatest impact on TimeKeeper CPU consumption is the time source accuracy. If a source is very noisy, inaccurate or shows high jitter TimeKeeper will apply extra filtering and use more data to establish a good time estimate which consumes more CPU. The more accurate a time source is the less CPU TimeKeeper requires.

CPU intensive options

Setting the LOWQUALITY=1 option will cause TimeKeeper to apply extra filtering and use more data for clock modeling. This uses quite a bit more CPU than the default of LOWQUALITY=0. Turning this option off will greatly reduce CPU use.

SOURCECHECK=1 will cause Timekeeper to cross-check all time sources and do a great deal more work than with this option disabled. Setting this value to the default of SOURCECHECK=0 will reduce CPU consumption.

Number of sources

The more sources that TimeKeeper must query, model and manage the more CPU TimeKeeper will use. Reducing the number of sources will reduce the amount of CPU that TimeKeeper consumes.

Query rate

Each time TimeKeeper queries a time source it must update the clock model and time estimates. This is where TimeKeeper consumes most CPU. Reducing the query rate can greatly reduce CPU consumption. For NTP sources this can be changed at the client side but for PTP sources must be configured on the server.

PTP vs NTP

PTP is a more complicated protocol than NTP and requires additional CPU in order to process. Using NTP instead of PTP can decrease CPU use slightly.
One useful feature of NTP is that the client defines the query rate rather than the server allowing multiple query rates from different clients using the same server.

LOWCPU option

The LOWCPU=1 option will cause TimeKeeper to use different clock models that require far less CPU than the default clock models. They are less accurate and less able to reject outliers. This option will make a significant difference in CPU consumption at the cost of a small amount of accuracy.

Network design

This section outlines some concepts and suggestions for designing a time sync network and general best practices. These recommendations are generic and meant to apply to a wide range of installations but of course some site to site variation will be necessary.

Below are two different sections describing different aspects of a time sync network design. The first deals with getting the best accuracy and most reliable time sync while the second focuses on monitoring, reporting and management concerns.

Time sync design

Simple is better

The best design philosophy for quality time sync is to keep things simple. We find that when doing initial setup/design of a network the first stumbling blocks are often because of enabling too many features in TimeKeeper, changing configurations from the defaults or making the network design too complicated. These problems can be avoided by allowing TimeKeeper to do what it defaults to in most cases.

The default values in TimeKeeper are designed to meet the requirements of most users and are based on years of helping ease deployment and testing. It's unlikely that you will need to change the default values when doing an initial TimeKeeper install to obtain a working setup, so we recommend you do not. Optimizing and tuning the configuration is something we recommend be done once a working setup and installation is complete.

Simple also means having as shallow a tree as possible for the time sync hierarchy. That's described below in more detail but a high level description would be to have no boundary clocks for PTP networks and as few (or no) stratum clocks as possible for NTP. That means ideally each client system talks directly to a top level time server connected to GPS or other high quality time source without intermediate nodes. Network traffic concerns, routing issues and time server load may require intermediate nodes, but the fewer the better as each introduces complexity and reduces the accuracy of time sync.

Separate network

A common question asked is whether a separate network is necessary for time sync. In general we say it is not. TimeKeeper is designed to handle the "jitter" and "noise" introduced by competition with other network traffic and can filter it out. A separate network for time sync would likely be expensive, time consuming to maintain and would only improve sync accuracy marginally if at all. TimeKeeper also makes very light use of the network so should not impact other operations. It typically sends and receives a single packet once per second.

Protocol choice - NTP or PTP

NTP and PTP, as implemented by TimeKeeper show equal accuracy. It is often mistakenly stated that NTP is less accurate than PTP because some NTP implementations are not as accurate as others. Some appliances use open-source implementations of the NTP protocol so it's best to ensure that your device has a high-quality and accurate implementation.

TimeKeeper provides equal accuracy for each protocol as it is a high-quality piece of software designed for accuracy. For additional information please see this whitepaper.

There are differences between NTP and PTP for monitoring and management. For more information please see the sections below.

Monitoring design

This section discusses some concepts when designing your time sync network for proper monitoring, logging and reporting. TimeKeeper is able to monitor client systems, share those logs with peer nodes for consolidation and alarm/alert/report when time is outside of range. The capabilities of TimeKeeper are greater than can be fully covered in this section so instead below we cover some best practices and general ideas when designing a time sync network.

TimeKeeper is able to both report accuracy information to other TimeKeeper and non-TimeKeeper monitoring systems and is able to query TimeKeeper and non-TimeKeeper hardware/software for time synchronization information. TimeKeeper is able to query, record and log information over many protocols and uses many techniques in order to log/collect as much information from a time synchronization network as possible.

NTP client monitoring

The NTP protocol has limited capability for monitoring. There are no protocol specific ways to query clients for detailed information. Instead the client provides it's estimated accuracy to the server when sending a time update request. This accuracy information is not very fine grained and is limited to 15 microsecond increments. That is, it can report 0 microseconds, 15 microseconds, 30 microseconds and similar. TimeKeeper will record and log this this information in $LOGDIR/timekeeperclients/ for each client.

PTP client monitoring

The PTP protocol provides more powerful and varied tools for client monitoring. As a result it is somewhat more complex. TimeKeeper attempts to use every available method to collect data from clients, server, boundary clocks and any other nodes on the timing network. A variety of mechanisms including TimeKeeper specific techniques and generic PTP protocol methods are used. Other PTP implementations may report partial or no information when queried. TimeKeeper clients will respond to both standard PTP management requests and also TimeKeeper specific queries.

The primary PTP monitoring method sends multicast and unicast PTP management requests to every reachable entity on the timing network (using unicast for environments where multicast is disabled or unavailable). These include standard PTP queries that collect a limited amount of information from non-TimeKeeper entities (if they respond). TimeKeeper-specific queries are also sent out on the network for more detailed and complete monitoring information.

TimeKeeper-specific queries will collect data from entities on the network for the last 15 seconds (the query interval period). This interval period can be changed in the TimeKeeper configuration using the MANAGEMENT_QUERY_INTERVAL directive in timekeeper.conf. TimeKeeper instances on the network will respond with the requested amount of data for its local time sources as well as any other entities that have responded to its queries. Responses to these queries will provide the IP address of the client systems and will not use hostnames.

TimeKeeper also sends standard PTP protocol queries. These are limited in the scope of information they collect and will only collect a single data point at a given instance in time rather than the much greater amount of information provided by TimeKeeper's other methods. Many PTP implementations (either hardware or software) do not respond to any queries at all, but some will respond to these minimal queries. TimeKeeper will record those in the same format as the above and report it. This method will use the IP address of the responding system and will not use hostnames.

Filesync

TimeKeeper versions 8.0.19 and later also use a method called "Filesync" by sending out queries to which TimeKeeper clients v8.0.19 and later will respond to over TCP port 319 (note that time sync occurs over UDP ports 319 and 320). Filesync responses include a complete set of information for the local system and any data stored in your configured $LOGDIR/timekeeperclients/*/ collected from remote entities on the network. This is the most robust monitoring method as it will recover from cases where a TimeKeeper instance has been disabled or unreachable for an extended period of time and needs to catch-up with any data it is missing.

The ENABLE_FILESYNC_QUERY setting will allow a host to send queries for Filesync client data on the network. ENABLE_FILESYNC_RESPONSE will allow a host to respond to any Filesync queries seen. Both of these options apply if the overall relevant ENABLE_MANAGEMENT_QUERY or ENABLE_MANAGEMENT_RESPONSE is also on. This means that if ENABLE_MANAGEMENT_QUERY is off, ENABLE_FILESYNC_QUERY will be forced off too, and the same applies the two response options.

Data collected/shared in this way will store data in your configured LOGDIR as $LOGDIR/timekeeperclients/[ipaddress]/* avoiding confusion with the other method which stores information in $LOGDIR/timekeeperclients/ipaddress_*.data*.

Note the Filesync method defaults to off.

Setting the management query window

Some deployments may require bulk transfers to happen at specific times of the day, for instance, after business hours. The times of the day during which TimeKeeper will query for data and respond to data requests using Filesync can be managed with the global configuration values FILESYNC_START_TIME and FILESYNC_END_TIME.

If set TimeKeeper will only request and respond to Filesync data requests between the defined start and end times respectively configured in “HH:MM:SS” format (hours, minutes, seconds) from the start of the day in UTC time.

For example, to enable queries and responses between 8:00 a.m. and 9:00 a.m. UTC as a global default, you would configure as follows in timekeeper.conf:

```
FILESYNC_START_TIME="08:00:00";
FILESYNC_END_TIME="09:00:00";
```

Note: TimeKeeper Compliance creates final audits for the previous UTC day 2 hours after UTC midnight. To ensure that client data is delivered to the Compliance host for audit generation on time, make sure to set your FILESYNC_START_TIME and FILESYNC_END_TIME values to ensure data delivery completes shortly after the end of the UTC day.

Restricting data shared

Some deployments may require that only specific clients' data is shared with remote servers. The parameters FILESYNC_INCLUDE and FILESYNC_EXCLUDE control which of the client systems in the log directory (default /var/log/timekeeperclients/*/) are shared with remote systems using the Filesync method. These parameters are set on the system that is hosting the files to allow or deny transmission of client data, meaning these options apply to the system with ENABLE_FILESYNC_RESPONSE set to send client data upstream. These fields use the following globbing patterns:

```
? matches any single digit
* matches zero or more digits
[] matches any single digit in the brackets or range of digits, e.g. [1-7]
```

For instance, the include specification:

```
FILESYNC_INCLUDE='???.?[1-7].12.\*';
```

Would match any clients in subnet 12 of any network with three major digits and 2 minor digits (like 192.17.12.122, but not 192.178.16.12) with a subnet value having any digit between 1 and 7 as its second place.

It would match,e.g.:

192.17.12.222

but would not match, e.g.:

10.22.12.2

To expand on the example, if you only wanted hosts with a host number of less than 100 on subnet 12 on those networks you could use:

```
FILESYNC_EXCLUDE='\*.???';
```

Which would prevent anything ending with a “.” followed by three digits from being included in the transferred files.

So 192.27.12.222 would no longer match, but 192.27.12.22 would.

Disable monitoring queries/responses

It is possible to disable responding to network PTP queries for information about sync accuracy as well as disable sending queries/recording the accuracy of other systems. This is sometimes done to reduce system and network load or when monitoring of specific clients is not necessary.

One can disable querying/recording time synchronization accuracy with:

```
ENABLE_MANAGEMENT_QUERY=0
```

Similarly one can disable responding to any queries seen with:

```
ENABLE_MANAGEMENT_RESPONSE=0
```

The Filesync method can be disabled by setting ENABLE_FILESYNC_QUERY=0 and ENABLE_FILESYNC_RESPONSE=0 in the timekeeper.conf file which is the default.

This can be done globally in the configuration file as well as on a per-source and per-server basis.

Timescales and how they apply to TimeKeeper

There are many timescales in timing (PTP, NTP, GPS, TAI, UTC, etc.) but most practical timekeeping is based on UTC (Coordinated Universal Time). Regardless of what timescale applies to the time delivered to TimeKeeper, it distributes time in UTC.

PTP and TAI/UTC

PTP may be distributed in different timescales, including the 2 common scenarios below which are of interest to TimeKeeper:

- TAI - the delivered time is based on the TAI timescale with an offset to be applied to transform the time to UTC
- UTC - the timescale is natively in UTC and there's no offset/correction needed

TimeKeeper can consume time according to either timescale and will convert to UTC as needed. When serving time it will deliver PTP in UTC. This means that clients getting PTP from TimeKeeper will get time with a reported 0 offset to UTC reported as it's already natively in that timescale.

This is rarely an issue although some clients that assume all PTP is delivered in TAI with a UTC offset. These clients may note/warn if the UTC offset is 0. In general these are just warnings but please contact the vendor for specific concerns. In some cases particular devices can get accurate time via NTP and optionally a PPS if they're unable to use UTC-based PTP directly.

UTC offset being 'reasonable' or traceable

In some configurations the timescale may be unrelated to any external reference, isn't matched to UTC in any way, or might be intended to be relevant to UTC but is no longer traceable to it.

PTP packets indicate whether the delivered time is reasonable relative to UTC. By default, if the PTP TimeKeeper sees doesn't indicate it's reasonable/traceable to UTC, TimeKeeper won't use that PTP and that TimeKeeper source won't be active. This is to protect against using invalid or untraceable sources of time.

Hoever, if this configuration is intended or expected, TimeKeeper can be configured to accept that PTP feed. Any TimeKeeper source that should accept PTP without being traceable to UTC should set ALLOW_UNREASONABLE_UTC=1 in the source definition. The rest of the configuration details are covered in the "Configuration" section.

TimeKeeper behavior on startup with seemingly invalid host time

TimeKeeper will check that the time on startup is not outside of a seemingly normal range. If the time is older than 30 years after the release of a given version of TimeKeeper or 5 years before then the system time will be set 5 years before the release date. This is to ensure proper startup of TimeKeeper when the time has been set to a wildly invalid time.

In some cases this means that on startup TimeKeeper will set the host time twice. First immediately on startup to make sure TimeKeeper can start correctly. Then once TimeKeeper has determined the correct time from a time source.

Incoming time validity checks

A similar check is applied to incoming time from all time source types. If the time is earlier than 5 years before or later than 30 years after the release of TimeKeeper the time sample will be rejected. This does mean that simulating time outside of what is considered reasonable time cannot be done. For information on how to do testing outside of these ranges please contact support@fsmlabs.com.

TimeKeeper Grandmasters

TimeKeeper is a software solution that users can run on their hardware according to their needs but it is also a set of hardware platforms and appliances.

In this section, we'll go over TimeKeeper Enterprise Grandmaster platform.

TimeKeeper Grandmasters

The TimeKeeper Grandmaster is the combination of the capabilities of TimeKeeper with a powerful hardware platform. In this section we'll go over what this combination lets you do, and also go over the Grandmaster features.

GPS/GNSS input

TimeKeeper Grandmasters will operate as stratum servers, boundary clocks, in addition to distributing time from GPS. In some deployments, a GPS/GNSS antenna may not be an option, or installation may still be in progress when the appliance is deployed.

By default, the primary time source is the GPS/GNSS input. If the GPS/GNSS isn't available, TimeKeeper will automatically fail over to any other defined sources so there's nothing that needs to be done. However, if you'd like to make sure the GPS/GNSS isn't used at all until the installation is complete, the GPS/GNSS source can be dragged to be the last source option (lowest priority) in the web GUI.

Note that the Grandmaster will retain a GPS/GNSS input, even if the GPS/GNSS source is deleted entirely. In that case, TimeKeeper will automatically define a GPS/GNSS source for you, but place it last in the list of sources.

GPS/GNSS connector

TimeKeeper Grandmaster Generation 2 uses a female SMA connection for the antenna input. This provides 5v to the antenna. The input to the device after amplifier and antenna gain as well as cabling/connector signal loss should be between 5 dB and 20 dB. Higher or lower than that can cause the system to not achieve GNSS lock. Cabling, antenna and connectors should all be 50 ohm.

PPS input/output

The default configuration for the PPS connector is as a PPS output. It can be optionally configured as an input instead. To configure this, connect the pulse-per-second signal to the SMA connection labeled "PPS" on the back of the Grandmaster. Then navigate to the *Configuration and Management* tab, select *Add a new source*, select *TimeKeeper PPS Card*, then uncheck *PPS Output*. Select the *Major time source* that should be used for wall-clock time then save and restart TimeKeeper.

PPS connector

TimeKeeper Grandmaster Generation 2 uses a female SMA 50 ohm connection for the PPS connector. The output is a 3.3v square wave with a duty cycle of 40%. When the configurable PPS input/output is configured for input a duty cycle of 20% or greater is recommended with a input pulse of 3.3v to 5v.

User management

The primary interface to the Grandmaster is via the web-based GUI. It is possible to access via ssh (once enabled), RS232 console, and keyboard/monitor. Some user accounts are disabled by default and must be explicitly enabled via the web GUI in order to be used.

These logins may be authenticated with RADIUS/TACACS+. See the "Grandmaster authentication - users and protocols" section for details.

VLAN and bonding support

VLAN interfaces and bonded interfaces are supported with TimeKeeper. Both are configured as normal for your distribution using the normal Linux tools. (VLAN configuration is also available on the TimeKeeper Grandmaster via the web GUI.)

Once the device is present, TimeKeeper can use it just like any other network device. A configuration that tracks PTP on VLAN 36 attached to eth0 would look like this:

```
SOURCE0() {
 PTPCLIENTVERSION=2
 IFACE=eth0.36
}
```

Similarly, a bonded interface can be named also. If the underlying device(s) behind the VLAN or bond support hardware timestamping, TimeKeeper will automatically use it.

RAID support

High availability storage is provided on the Grandmaster out of the box. This storage is monitored by TimeKeeper and alerted on as needed. For details on managing the RAID devices, please refer to the "RAID alerting and recovery" section.

Factory reset

The factory settings for network configuration and time service settings can be restored with a "factory reset." To perform the factory reset you must log into the device via RS232 console (using 115200 baud, 8n1 or 57600 baud on older models) or through a connected keyboard/monitor. If you are connecting a laptop directly to the RS232 port, you must use a null-modem cable. Log in with the account "factoryreset" and the password "timekeeper". Note: In earlier versions of TimeKeeper the password was "fsmlabs". Once that is done you will be asked to confirm twice that you wish to perform a factory reset, and whether you'd like to shut down the unit when the step is complete. Once that is done the settings will be restored and the system will reboot or shutdown as requested. Once booted the system is back to factory original settings.

Command line configuration

Some Grandmaster capabilities, like networking options, can be configured from the command line. This allows users to connect to a system via the console remotely and easily configure the system so that it's accessible via the rest of the network.

In general, once the network is configured, much of the remaining configuration is done via the web interface, either over http or https.

To use the command line interface, login as the "admin" user, (detailed in the section, "Authentication - users and protocols"), via keyboard/monitor, RS232 console, or ssh. Once logged in, run *timekeeper_cli* from the shell and follow the prompts.

Some of the options offered by the command line interface involve editing files. In those cases, the environment variable *EDITOR* guides which editor will be used. Editor options include *vi* (the default) and *nano*, for example. To use the nano editor, run this command before executing *timekeeper_cli*:

```
export EDITOR=nano
```

Advanced routing options

The web interface can be used to set up most networking configurations on the Grandmaster, including interface addresses, the default gateway, DNS settings, VLANs, bonded interfaces, and more.

However, some networks require more complex routing capabilities. Policy based routing is one example of this. For users that do require more advanced capabilities, the TimeKeeper CLI mentioned above can be used to configure the device.

To configure these options, login as the *admin* user on the console or via SSH, and run *timekeeper_cli*. This will give you an option to configure advanced networking capabilities, including per-device routes and routing tables.

UDP and IGMP

TimeKeeper operates using the NTP, Time and PTP protocols using the standard assigned ports in each case. Because the Grandmaster has multiple network interfaces, it may be necessary to adjust your network's routing to allow and/or correctly route these packets.

The TimeKeeper Grandmaster allows access via the timekeeper_cli command described earlier to editing Linux's standard "/etc/sysconfig/network-scripts/route-device" files where device is the name of the interface on which the route is being added.

For instance, if you had a system or network which needed to be accessed other than by the default route, you would enter the timekeeper_cli application, and apply changes by selecting:

```
[admin@gm ~]$ timekeeper_cli
...
2. Configure advanced routing options
...
> 2
...
3. Configure per-device routes
...
> 3
```

Then selecting the device which may require advanced routing configuration and editing its configuration there, using the standard format for Linux per-device routes, e.g.:

```
224.0.1.0/24 via 192.168.0.1 dev eth0
```

with the format 'NETWORK/NETMASK "via" ROUTER "dev" DEVICE'. The above would enable broadcast UDP to be sent to the 224.0.1.0/24 network via a router at 192.168.0.1 on device eth0.

In uncommon scenarios, packets destined for an address may not have a route to a device. In those cases, like with broadcast PTP (224.0.1.129), you may need to configure a route specfically for that device.

TimeKeeper will send multicast UDP packets on a configured interface for a given source or server regardless of routing policies.

If you're routing multicast it may also be necessary to add routes to your multicast routers using the method above in addition to enabling Protocol Independent Multicast (PIM) or a variety of IGMP switch management features which are beyond the scope of this documentation. For information on IGMP snooping and routing multicast UDP, please consult your router and/or switch documentation. Alternatively, if multicast routing becomes complex, TimeKeeper client and server can instead use unicast PTP or NTP.

If you have questions on whether this option applies to you, please contact support@fsmlabs.com.

TimeKeeper Grandmaster

This section of the Grandmaster documentation will just focus on managing the device - from initial setup to applying updates. For data on individual features, please refer to the section on that feature.

TimeKeeper Grandmaster setup

Connect a GPS antenna to the female SMA GPS (Grandmaster generation 2) or BNC (Grandmaster generation 1) connector on the back of the device (see photos below). The GPS receiver provides 5v DC to the antenna.

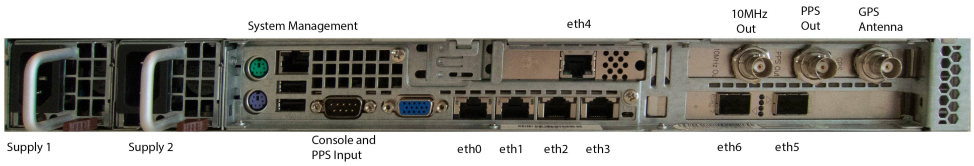
Connect a network cable to the first RJ45 ethernet port (eth0 in photos) and navigate with a web browser to 192.168.1.1. You can log in with account “admin” and password “timekeeper” (Note: In earlier versions of TimeKeeper the default password was “fsmlabs”) to configure the network ports, TimeKeeper mode and options. Change the admin user’s default password immediately for security, please. If this is an upgrade and you have been using previous versions of the Grandmaster GUI, it is recommended that you force a full page reload (Ctrl-Shift-R on many browsers). This will make sure your browser loads the latest web interface from the Grandmaster and provides all of the latest features.

To change network interface addresses navigate to the *Configuration* tab and then *Networking* subtab, select the options you would like then click *Save network changes* and then *Restart Network* to apply the updates. The changes will take a moment to apply after which you can reconnect to the newly configured address.

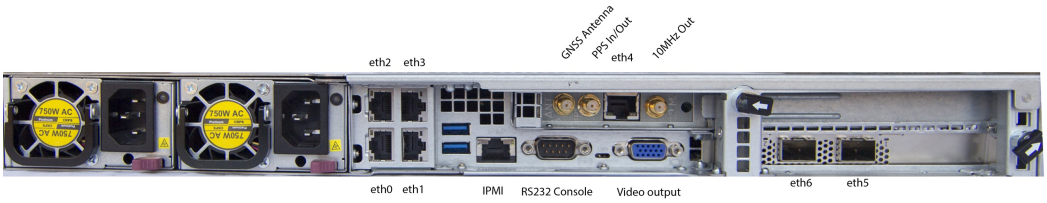
By default (and as shipped) the Grandmaster does not enable SSH access. That can be enabled through the web interface, however. Please see below for details on enabling additional access methods.

The device will then synchronize with the GPS and once done will answer NTP queries and provide PTPv2 service on the default domain (domain 0) on all network interfaces. The GPS startup process can take up to 15 minutes in some cases. You can monitor progress on initialization via the *Status* tab in the web GUI.

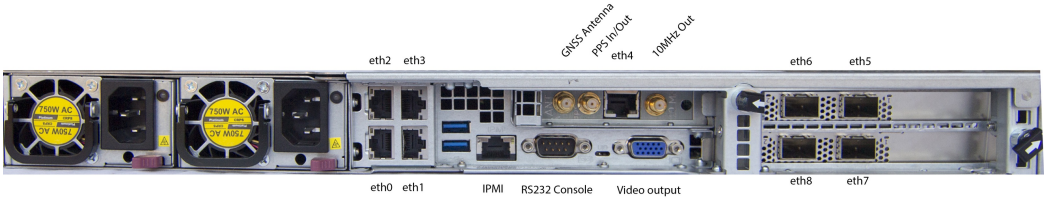
No further configuration of the server should be necessary.



Grandmaster Generation 1



Grandmaster Generation 2



Grandmaster Generation 2 with Additional Network Interfaces

Shipped network configuration

Each network port is configured with a unique static IP address when shipped. eth0 is configured for 192.168.1.1, eth1 for 192.168.2.1, and so on. This includes optional 10G SFP+ ports which are eth5 (192.168.6.1), eth6 (192.168.7.1), etc.

Please note, the TimeKeeper Grandmaster does not support having multiple network interfaces on the same subnet or multipath routing, and does not support netmasks more restrictive than /30 (255.255.255.252) to avoid common routing issues.

Modifying the server configuration by changing the console ports, otherwise modifying the Linux kernel, BIOS settings, installed devices or installed software applications may cause problems in performance or correctness. This can create support issues beyond FSMLabs' support obligations. Please contact support@fsmlabs.com before attempting to make any changes to the server software or hardware configuration.

Upgrading TimeKeeper Grandmaster hardware

The following steps describe how to upgrade via the web interface. In addition to TimeKeeper, you can also upgrade the *basepackage*, which is a set of components that upgrade the Grandmaster platform and provide new features for TimeKeeper to use. Please note, however, that when updating TimeKeeper and the basepackage on a hardware Grandmaster, TimeKeeper must be upgraded first, followed by the basepackage.

1. Download the installer for the current release from FSMLab's website
2. Log in to the web interface (see the "Web interface authentication" section for details)
3. Click on the "Configuration tab" on your GM
4. Click on the "Update" tab under that
5. Click "Update TimeKeeper install"
6. Click "Choose File"
7. Navigate to and select the installer you downloaded in step 1
8. Wait about 20 minutes while the system reboots twice to complete the upgrade process
9. Log back in to the web interface
10. Confirm via the "Status" tab that it has the right version
11. If you also want to install the basepackage, repeat except click "Update TimeKeeper base package" in step 5

RAID alerting and recovery

TimeKeeper Grandmasters have built in RAID support, and the state of the RAID will be reported on the status page of the TimeKeeper web GUI. If configured, TimeKeeper will also send alarms about disk failure and recovery through the normal configured alerting methods - syslog, email, SNMP, detailed in the section, "Alerting - Log files, SNMP, syslog, email, Windows event log." Alarms will also be sent if the S.M.A.R.T. diagnostics of a disk reports that it may fail.

In the event of a disk failure, the Grandmaster will emit periodic alarms until the failed disk is replaced, but will continue operating. Once the failed disk is replaced, a reboot is required in order to initiate reconstruction of the RAID array.

Managing Grandmaster systems

TimeKeeper Grandmaster systems can have some of their settings saved, set, and restored with the `tkctl` tool. tkctl can also be used to change individual settings directly on the command line.

To use tkctl log in on the console or through ssh:

```
tkctl -h
```

All current Grandmaster settings can be printed with this command, as the *admin* user:

```
tkctl -a
```

Redirect this output to a file to save the current grandmaster settings:

```
tkctl -a > /tmp/tkgm_settings.tkctl
```

This file can be saved externally as a backup of the system state and modified if needed to reflect intended settings. This file can be applied with the -l option:

```
tkctl -l /tmp/tkgm_settings.tkctl
```

Settings are applied in the order they're defined in the file. For example, if a line specifies the IP address of a VLAN interface, that VLAN interface must already be created so any line needed to create the device should be specified first. More detail on device creation is listed below.

These steps of saving off the configuration and applying the saved configuration can also be done with the web interface, under the *Configuration* tab, subtab *Service and System Management*. The section *Configuration Management* has an option to download and save the current configuration and also one to upload an existing configuration file to have it applied.

Individual settings can also be set or retrieved, like this to get the IP address set for eth0:

```
tkctl -g gm.network.devices.eth0.ipaddr
```

or to set it:

```
tkctl -s gm.network.devices.eth0.ipaddr=10.45.220.31
```

tkctl options

`tkctl` controls 2 different types of behavior on TimeKeeper Grandmasters - settings and actions. Settings are individual feature configurations like network options. Actions are steps involving an action and may relate to a setting, like restarting the network to apply configuration changes recently made. Here's an example where you set the configuration for eth0, then restart the network in order to make that setting active:

```
tkctl -s gm.network.devices.eth0.ipaddr=10.45.220.31
tkctl -s gm.network.devices.eth0.netmask=255.255.255.0
tkctl -s gmaction.restart=gm.services.network
```

Or similarly a bond device can be created/configured/deleted:

```
tkctl -s gmaction.create=gm.network.devices.bond0
tkctl -s gm.network.devices.bond0.ipaddr=10.45.220.31
tkctl -s gm.network.devices.bond0.netmask=255.255.255.0
tkctl -s gm.network.devices.eth5.slave=1
tkctl -s gm.network.devices.eth5.master=bond0
tkctl -s gm.network.devices.eth7.slave=1
tkctl -s gm.network.devices.eth7.master=bond0
tkctl -s gmaction.delete=gm.network.devices.bond0
```

Here are the list of names/settings supported by `tkctl` and can be set (with -s) if applicable or retrieved (with -g). A setting that isn't present or doesn't have a value set may not be listed when using `tkctl -a`. If `tkctl -g` is used to query a setting that isn't present, an empty value may be printed in addition to the expected values specified below.

- gm.type - The type of GM (readonly).
- gm.serial - The serial number of the GM (readonly).
- gm.config.timekeeper[1] - The TimeKeeper config file to be used, specifically the /etc/timekeeper.conf contents, base64 encoded.
- gm.config.ntpkeys[2] - The NTP keys to be used, base64 encoded.
- gm.config.relayhost[2] - A host to be used to relay email alerts through.

- gm.config.maxlogage[2] - A numeric value indicating how many days to retain logs for.
- gm.services.sshd.enabled[2] - 1 or 0, indicates whether SSH is enabled for remote login.
- gm.services.sshd.listenaddress.{1,2,3}[2 3] - An IP address that should answer SSH queries.
- gm.services.snmpd.enabled[2] - 1 or 0, indicates whether SNMP querying is enabled.
- gm.services.snmpd.communitystring[2] - A base64 encoded string representing the community string to be used for SNMP queries.
- gm.services.syslog.{1,2,3}.host[2 3] - Hostname/IP address of the first, second, or third configured remote syslog host.
- gm.services.syslog.{1,2,3}.port[2] - The numeric port to be used for remote syslog communication for the first, second, or third server.
- gm.services.syslog.{1,2,3}.protocol[2] - The protocol used to reach the specified remote syslog server - TCP or UDP.
- gm.auth.tacacs.enabled[2] - 1 or 0, indicates whether TACACS+ is enabled for authentication (if present).
- gm.auth.tacacs.{1,2,3}.host[2 3] - Hostname/IP address of the first, second, or third configured TACACS+ server.
- gm.auth.tacacs.{1,2,3}.secret[2] - A base64 encoded secret to be used to communicate with the identified TACACS+ server. This can only be set, not retrieved, and is recommended to be loaded from a file rather than set interactively.
- gm.auth.radius.enabled[2] - 1 or 0, indicates whether RADIUS is enabled for authentication (if present).
- gm.auth.radius.{1,2,3}.host[2 3] - Hostname/IP address of the first, second, or third configured RADIUS server.
- gm.auth.radius.{1,2,3}.secret[2] - A base64 encoded secret to be used to communicate with the identified RADIUS server. This can only be set, not retrieved, and is recommended to be loaded from a file rather than set interactively.
- gm.auth.radius.{1,2,3}.timeout[2] - Timeout in seconds for the first, second, or third configured RADIUS server.
- gm.users.root.enabled[2] - 1 or 0, indicating whether the root user can log in via SSH.
- gm.users.{admin,readonly,loguser}.password[2] - A base64 encoded password to be used for the user. This can only be set, not retrieved, and is recommended to be loaded from a file rather than set interactively.
- gm.users.{readonly,loguser}.enabled[2] - 1 or 0, indicating whether the user is enabled. The user must have a password set in order to be enabled.
- gm.network.hostname[4] - The current hostname for the grandmaster.
- gm.network.dns.0[4 3] - The primary DNS server the grandmaster will use.
- gm.network.dns.1[4 3] - The secondary DNS server the grandmaster will use.
- gm.network.gateway[4] - The default gateway the grandmaster will use.
- gm.network.rt_tables[4] - A base64 encoded string representing the global route table names.
- gm.network.devices.ethX[4] - A list of settings relevant to each ethernet device present. The number of entries will vary by configuration. Valid settings include:
  - gm.network.devices.ethX.enabled - 1 or 0, indicates whether the device will be used or not.
  - gm.network.devices.ethX.dhcp - 1 or 0, indicates whether the device will get address details through DHCP.
  - gm.network.devices.ethX.ipaddr - A valid IP address to be used for the device.
  - gm.network.devices.ethX.netmask - A valid network mask to be used for the device.
  - gm.network.devices.ethX.slave - 1 or 0, set to 1 if it's a member of a bonded interface.
  - gm.network.devices.ethX.master - The name of the bond device this device is a member of, if relevant. Requires ethX.slave=1 to previously be set.
  - gm.network.devices.ethX.descr - A string used to describe the interface for easier management.
- gm.network.devices.bondX[4] - A list of settings relevant to each bond device present. The number of entries will vary by configuration. Valid settings include:
  - gm.network.devices.bondX.enabled - 1 or 0, indicates whether the device will be used or not.
  - gm.network.devices.bondX.dhcp - 1 or 0, indicates whether the device will get address details through DHCP.
  - gm.network.devices.bondX.ipaddr - A valid IP address to be used for the device.
  - gm.network.devices.bondX.netmask - A valid network mask to be used for the device.
  - gm.network.devices.bondX.descr - A string used to describe the interface for easier management.
- gm.network.devices.devX/Y[4] - A list of settings relevant to each VLAN device present for VLAN Y, tied to a specific ethernet or bond interface (devX). The number of entries will vary by configuration. Valid settings include:
  - gm.network.devices.devX/Y.enabled - 1 or 0, indicates whether the device will be used or not.
  - gm.network.devices.devX/Y.dhcp - 1 or 0, indicates whether the device will get address details through DHCP.
  - gm.network.devices.devX/Y.ipaddr - A valid IP address to be used for the device.
  - gm.network.devices.devX/Y.netmask - A valid network mask to be used for the device.
  - gm.network.devices.devX/Y.descr - A string used to describe the interface for easier management.
  - gm.network.devices.devX/Y.rule - A base64 encoded string representing the device specific Linux rules for this interface.
  - gm.network.devices.devX/Y.route - A base64 encoded string representing the device specific Linux routes for this interface.
- gm.network.devices.ipmi[4] - A list of settings relevant to the IPMI. Valid settings include:
  - gm.network.devices.ipmi.dhcp - 1 or 0, indicates whether the IPMI will get address details through DHCP.
  - gm.network.devices.ipmi.ipaddr - A valid IP address to be used for the IPMI.
  - gm.network.devices.ipmi.netmask - A valid network mask to be used for the IPMI.
  - gm.network.devices.ipmi.gateway - The default gateway the grandmaster will use for the IPMI.

A number of options can be taken using a 'set' using the gm.delete name to remove a name/setting. This can be done as above with a command like:

```
tkctl -s gmaction.delete=X
```

where X is the value to be cleared or removed. Names that allow deletions include:

- gm.services.sshd.listenaddress.{0,1,2}[3] - This deletes the specified address that SSH is configured to accept queries on.

- gm.services.syslog.{1.2.3}.host[3] - 1, 2, or 3 indicates the configured syslog server entry to delete. Similarly, the syslog entry can be removed by setting the syslog host's value to an empty string.
- gm.auth.tacacs.{1.2.3}.host[3] - 1, 2, or 3 indicates the configured TACACS+ server entry to delete. Similarly, the TACACS+ entry can be removed (including the configured secret) by setting the entry's host value to an empty string.
- gm.auth.radius.{1.2.3}.host[3] - 1, 2, or 3 indicates the configured RADIUS server entry to delete. Similarly, the RADIUS entry can be removed (including the configured secret and timeout) by setting the entry's host value to an empty string.
- gm.network.devices.bond - Where the bond named is bond0, bond1, bond2, etc. This deletes a configured bond device
- gm.network.devices.ethX/YY - Where the ethernet/VLAN device named is eth3/44 for VLAN 44 on eth3 for example. This deletes the configured VLAN on the named ethernet device.
- gm.network.devices.dev.rule - This deletes the configured custom Linux rules for the named device.
- gm.network.devices.dev.route - This deletes the configured custom Linux routes for the named device.
- gm.network.dns.{0,1}[3] - This deletes the configured first or second DNS server.

Here are a list of sample actions possible:

- gmaction.restart=gm.services.timekeeper - Restarts TimeKeeper only.
- gmaction.restart=gm.services.network - Restarts the network (and TimeKeeper).
- gmaction.restart=gm.services.syslog - Restarts the syslog service.
- gmaction.restart=gm.services.sshd - Restarts the sshd service.
- gmaction.restart=gm.hardware - Restarts/reboots the Grandmaster.
- gmaction.powercycle=gm.hardware - Hard power cycles the Grandmaster. This should be used after any GPS/GNSS antenna connections have been modified or disconnected and reconnected.
- gmaction.stop=gm.services.timekeeper - Stops the TimeKeeper service temporarily.
- gmaction.stop=gm.hardware - Shuts the Grandmaster down and powers it off.
- gmaction.delete=gm.services.sshd.listenaddress.3 - Delete the 3rd configured ssh listen address.
- gmaction.create=X - Creates named device, either a bond or VLAN as described above. An example is gmaction.create=gm.network.devices.eth2/33 to create VLAN device eth2.33 tied to eth2. Does not report an error if the device already exists.
- gmaction.delete=X - Deletes named device, either a bond or VLAN as described above. An example is gmaction.delete=gm.network.devices.eth2/33 to delete VLAN device eth2.33 on eth2.
- gmaction.trigger=testalert - Causes TimeKeeper to log and send a test alert over all configured mechanisms (SNMP, syslog, email).

These operations are reported over syslog along with some additional events in order to provide records of changes made over time by different users. Please refer to the alerting documentation for details on what these alerts look like and how they're transmitted.

Note: `tkctl -a` does not include deletions of any deletable settings (see gmaction.delete above) including SSH listen addresses, syslog servers, RADIUS/TACACS+ servers, bonds, VLANs, routes, and rules. That means, upon using `tkctl -l` to load a backup saved with `tkctl -a`, you will end up with a *union* of the old and new configurations with respect to these settings. If you don't want this union, be sure to delete these settings before restoring from the backup file. You can do this from the web interface or with the `tkctl` command, e.g.,

```
tkctl -s gmaction.delete=gm.auth.radius.3.host
tkctl -s gmaction.delete=gm.auth.radius.2.host
tkctl -s gmaction.delete=gm.auth.radius.1.host
```

Similarly, `tkctl -a` includes settings for any configured bonds and VLANs, e.g., `gm.network.devices.bond0/1.enabled = 1`, but it does not first create those bonds or VLANs. To prevent `tkctl -l` from failing, be sure to first create any bonds and VLANs referenced in the backup file before loading it. You can do that from the web interface, via the `tkctl -s` command, or by adding creation settings to the configuration file, e.g.,

```
gmaction.create=gm.network.devices.bond0
gmaction.create=gm.network.devices.bond0/1
```

for the above case.

Quick steps for Grandmaster replacement

A common use of `tkctl` is to save and restore entire Grandmaster configurations. Saving the configuration is important to provide continual backups of the entire Grandmaster configuration in case of failure. Restoring the configuration with `tkctl` is helpful when there's been an inadvertent change, or when an existing Grandmaster needs to be replaced with a new unit running the same configuration.

To recap, a Grandmaster's current configuration can be saved off in a file with:

```
tkctl -a > /tmp/gm_config
```

from the command line, or from the web interface select *Configuration* tab, subtab *Service and System Management* and click *Save Grandmaster configuration* to download the current settings as a file.

Once this file is in hand it can be applied to a new GM to get it up and running as a replacement in production, or as an identically configured unit to be used in test. This is done at the command line with:

```
tkctl -l /tmp/gm_config
```

from the command line, or from the web interface select *Configuration* tab, subtab *Service and System Management* and click *Apply Grandmaster configuration file* to upload and apply the existing file.

When using the web interface, the upload will also trigger a restart of both the network and TimeKeeper automatically. The `tkctl` command line interface will leave that step for the user. To make sure all settings are applied follow the `tkctl` load step above with:

```
tkctl -s gmaction.restart=gm.services.network
```

Alternatively, the provided configuration file could have *gmaction.restart=gm.services.network* included as the last line to automatically restart both the network and TimeKeeper.

TimeKeeper Grandmaster quick start

Quick setup

Connect a GPS antenna to the female SMA GPS (Grandmaster generation 2) or BNC (Grandmaster generation 1) connector on the back of the device (see photos below). The GPS receiver provides 5v DC to the antenna.

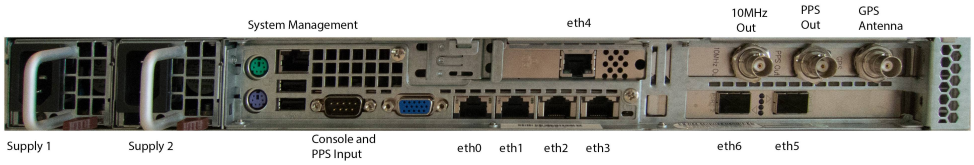
Connect a network cable to the first RJ45 ethernet port (eth0 in photos) and navigate with a web browser to 192.168.1.1. You can log in with account "admin" and password "timekeeper" (Note: In earlier shipped Grandmasters the default password was "fsmlabs") to configure the network ports, TimeKeeper mode and options. Change the admin user's default password immediately for security, please. If this is an upgrade and you have been using previous versions of the grandmaster GUI interface, it is recommended that you force a full page reload (Ctrl-Shift-R on many browsers). This will make sure your browser loads the latest web interface from the grandmaster and provides all of the latest features.

To change network interface addresses navigate to the "Configuration" tab and then "Networking" subtab, select the options you would like then click "Save Network changes" and then "Restart Network" to apply the updates. The changes will take a moment to apply after which you can reconnect to the newly configured address.

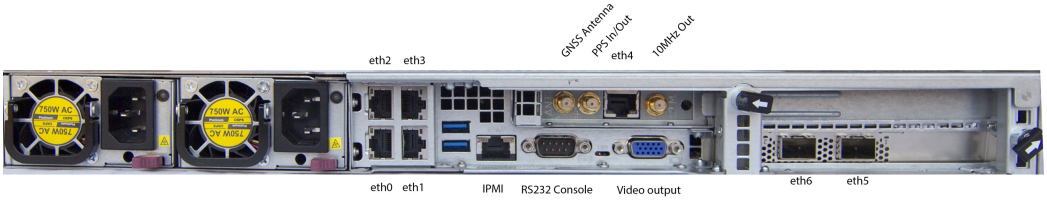
By default (and as shipped) the grandmaster does not support SSH access for configuration. That can be enabled through the web interface, however. Please see below for details on enabling additional access methods.

The device will then synchronize with the GPS and once done will answer NTP queries and provide PTPv2 service on the default domain (domain 0) on all network interfaces. The GPS startup process can take up to 15 minutes in some cases. You can monitor progress on initialization via the "Status" tab in the web GUI.

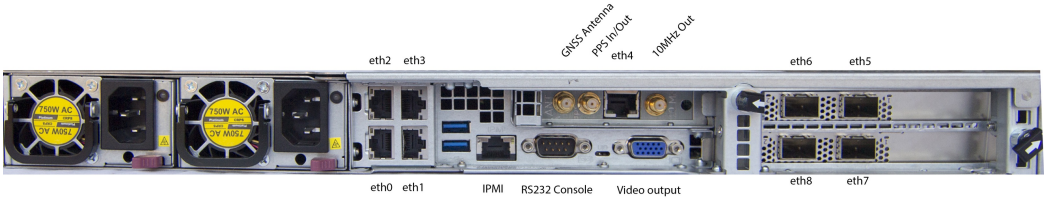
No further configuration of the server should be necessary.



Grandmaster Generation 1



Grandmaster Generation 2



Grandmaster Generation 2 with Additional Network Interfaces

Shipped network configuration

Each network port is configured with a unique static IP address when shipped. eth0 is configured for 192.168.1.1, eth1 for 192.168.2.1 and so on. This includes optional 10G SFP+ ports/25G SFP28 which are eth5 (192.168.6.1) and eth6 (192.168.7.1) and so on.

Please note, the TimeKeeper Grandmaster does not support having multiple network interfaces on the same subnet or multipath routing.

Remote Control and Management (IPMI)

The IPMI functions are provided by the Baseboard Management Controller (BMC) which starts as soon as power is connected to the system. The dedicated IPMI network interface is configured to use DHCP when shipped. The default username is “ADMIN”.

For improved cybersecurity compliance, the default IPMI passwords are now unique for each chassis. On newer Grandmasters built and shipped December 2022 onwards (loaded with Timekeeper 8.0.27 or later) the password is a unique alphanumeric code labeled as ‘PWD: ##########’ found on the bottom (under) side of the physical pull-out Service Tag. The Service Tag is a retractable plastic ‘tongue’ that is located just above the leftmost drive tray when viewing the front panel.

Service Tag Retracted

Top of Service Tag

Bottom of Service Tag (password is the PWD field)

Earlier versions of Grandmasters use the default password “ADMIN”.

Factory reset

The factory settings for network configuration and time service settings can be restored with a “factory reset.” To perform the factory reset you must log into the device via RS232 console (using 115200 baud, 8n1 or 57600 baud on older models) or through a connected keyboard/monitor. If you are connecting a laptop directly to the RS232 port, you must use a null-modem cable. Log in with the account “factoryreset” and the password “timekeeper”. Note: In earlier versions of TimeKeeper the password was “fsmlabs”. Once that is done you will be asked to confirm twice that you wish to perform a factory reset. Once that is done the settings will be restored and the system will reboot. Once booted the system is back to factory original settings.

TimeKeeper patents

FSMLabs TimeKeeper, TimeKeeper Grandmaster and other products are protected under US and Foreign Patents. Patent info at

https://www.fsmtime.com/about/patents/

TimeKeeper Version 8

TimeKeeper Version 8.0.x

v8.0.31 - February 14, 2024

- Add tkctl action for the Grandmaster to trigger SNMP/syslog/email events on demand for testing and connectivity verification.
- Improve startup sync by detecting when other tools have stepped the clock during startup, particularly during system boot.
- Improve file handling under high load on Windows.
- Remove support for Windows Server 2012.
- If configuration file contains DOS newlines on non-Windows hosts, output error message and exit.
- Remove dependency on ifconfig for some NTP server configurations on Linux.
- For NICs with more interfaces than PHCs, track PHC ownership internally by the interface with sysfs PHC files present, rather than by other metrics.
- Improvements to encodepassphrase tool in cases where certain passphrase lengths were not being successfully decrypted.
- Identify source when mouse cursor hovers over a client in the clients Live Graph legend and add source-identity column to client Snapshot View table.

v8.0.30 - November 11, 2023

- Fix to allow Compliance to continue PDF generation when SQL database is corrupted.
- Update packaged Node.js to version 16.20.2.
- Update Node.js modules formidable to 3.5.1 and express to 4.18.2.
- Ensure PHCs are attached to root devices and not interface aliases.
- Add ability to send test alerts from the web interface on non-grandmasters.
- Add support for RHEL 8 specific RPMs, including SHA256 digests.
- Add security enhancements regarding HTTPS headers and hyperlinks.
- Upgraded Bootstrap to v5.3.2 for documentation.

v8.0.29 - September 25, 2023

- Add DISABLE_CLOCK_STEER that prevents TimeKeeper from steering clocks, including the system clock.
- Remove unused Node.js modules, including jquery, cryptiles, extend, jsonpointer, json-schema and tough-cookie.
- Have tkctl warn when setting a user account to enabled with an empty password.
- Add official support for Amazon Linux 2023.
- Preserve AAA configuration across TimeKeeper installations on first-generation Grandmasters.
- Improved the input validation in the web interface for Compliance report parameters and PTP/NTP servers.
- Upgraded components used by the TimeKeeper documentation web interface for increased security.
- Add ENABLE_PAM_AUTH to authenticate a web interface user with the PAM user name and password from the Linux host. This feature enables LDAP authentication in the web interface on Linux.
- Include FQDN in syslog messages rather than just hostname.

v8.0.28 - May 19, 2023

- Fixed a vulnerability that could allow remote code execution via the web interface if a valid login session and a specifically formed URL are supplied.
- Fixed an issue where certain scenarios can cause the web UI login page to hang with status “Authenticating…”.
- Fixed an issue that prevented saving large configurations in the web interface.
- Fixed an issue that prevented removal of network interfaces from bonds in the web interface.
- Add ability to hard power cycle a Grandmaster from the web interface, tkctl and timekeeper_cli.
- Changed default IPMI passwords for new Gen 2 Grandmaster to be unique per appliance, please see Quick Start for more information.
- Enable serial console access on new Gen 2 Grandmaster.

v8.0.27 - November 9, 2022

- If a source becomes erratic and triggers LOWCPU mode for steering, ensure it returns to normal processing once it stabilizes.
- Fix bug when downloading active series as CSV from web interface by first enclosing fields containing commas in quotes.
- Allow for faster rejection of hardware timestamps in cases where NIC drivers behave erratically.
- Fixed issue to allow PPS output when GPS antenna is not connected.
- Resolve login issue with TimeKeeper Client or Server, when ‘%’ used in web GUI passwords.
- Increase the timeout allowed for web logins for cases where RADIUS/TACACS+ servers are in use but unavailable and local authentication is needed.
- Fix to allow Interface Description to be saved correctly in web UI.
- Fixed issue on Grandmasters that caused hostname to revert to ‘fsm’ on reboot.
- Fix bug where TimeKeeper was accepting a unicast announce for a multicast source.

v8.0.26 - August 18, 2022

- Upgrades to jQuery, jQuery UI versions used for the TimeKeeper web interface. This also includes minor usability improvements.
- Remove support for Internet Explorer (now EOL) when using the web interface. Microsoft Edge and other major browsers continue to be supported.
- Improve PDF viewing of Compliance reports on web interface.
- Use Form-Based Authentication for web GUI instead of Basic Authentication.
- TimeKeeper web interface now delays repeated invalid login attempts.
- Fix for the Compliance tab where reports were not displayed when a large number of Compliance reports exist.

- Reduce CPU load when handling high numbers of PTP/NTP requests and increase PTP/NTP serving performance.
- Recover from very rare case where invalid network delay measurements could prevent proper clock steering.
- TimeKeeper CPU load reduced by ensuring more consistent hardware TX timestamps.
- Fixed bug that would allow Sourcecheck to reject the last active source if it looks invalid rather than retaining it (introduced in 8.0.23).
- Remove dependency on host openssl configuration when encrypting SNMP passphrase.
- Compliance generate utility provided to allow Compliance reports to be generated outside TimeKeeper. See the compliance_generate documentation for details.
- Add support for the automotive PTP profile.
- Fix to retain the specified web management port in timekeeper.conf when set to 8080, now that the default value is -1 (introduced in 8.0.25).

v8.0.25 - April 29, 2022

- Web GUI now defaults to HTTPS using provided self-signed certificate. HTTP defaults to off unless WEB_MANAGEMENT_PORT is specified and not set to -1.
- Add support for tracking revision history and events on TimeKeeper Grandmasters and software installations. See the alerting documentation for details.
- Improve accuracy by eliminating oscillations that can occur with lower accuracy time sources.
- Direct support for Hyper-V provided time on Linux where available, such as in Azure cloud environments.
- Detect and alert when sustained levels of GNSS jamming indicates that holdover accuracy may be reduced, should it be needed.
- Add plot for TimeKeeper Grandmaster satellite, jamming, and signal strength data on TimeKeeper Grandmasters and clients that collect their management data.
- Add support for Layer 2 PTP on Linux.
- Fixes spacing on Compliance report PDF headers.
- Fix for saving large configuration files through the web interface with Node.js version 14.16.0.
- Fix for building Compliance reports for clients with high update rates and rare accumulated log file sizes.
- Fix to prevent TimeKeeper Grandmasters from using software rather than hardware timestamps on 1G interfaces in some scenarios.
- Remove support for SLES 11.
- Add official support for Windows Server 2022.

v8.0.24 - January 28, 2022

- Handle the case of NTP servers that do not respond to all requests more gracefully.
- Faster and more accurate slewing during initial startup with large offset for greater accuracy sooner.
- High accuracy sync happens much faster when hardware timestamps are being used.
- Update for TimeKeeper Grandmasters to address polkit's pkexec vulnerability (CVE-2021-4034).

v8.0.23 - December 13, 2021

- Support TLS 1.3 on the web interface where it's available using the TLS_VERSION option.
- Fix bug displaying client data in Timing Quality live graphs in web interface, introduced in 8.0.22.
- Enable hardware and software timestamping for Windows platforms that support it (Windows Server 2022 and Windows 11 or later).
- Add WEB_MANAGEMENT_TIMEOUT to support a configurable session timeout for the TimeKeeper web interface.
- Add support for an unlimited number of network interfaces.
- Allow recovery and use of sources that have become otherwise unusable, for some very rare conditions.
- When using Sourcecheck allow for a faster failover between sources when one is found to be invalid. This retains better accuracy with remaining valid sources.
- Add to the list of known-unstable driver versions TimeKeeper automatically avoids to avoid causing device and kernel problems.
- Fix issue when updating license in place during downtime window so that it no longer requires a TimeKeeper restart.
- Update default password for admin account to "timekeeper", and deprecate the old password "fsmlabs". Note this new default will apply even if a downgrade is performed.
- Allow for more PTP multicast group memberships across more devices on TimeKeeper Grandmasters.
- When saving a configuration change via the web interface, don't write options to timekeeper.conf that have default values or are otherwise redundant.

v8.0.22 - October 6, 2021

- Add INACTIVE_CLIENT_THRESHOLD to activate an alarm if a client is detected as inactive.
- Display Grandmaster type on status tab of web interface.
- Prevent hot-plug PHC device removal from potentially affecting processing of timing data.
- Deprecated support for the prebuilt TimeKeeper RPMs on SLES 11. SLES 11 remains a supported distribution, including installations using local RPMs built with the TimeKeeper .spec files.
- Provide a warning on the web interface if /etc/timekeeper.conf is not present/world readable on Linux as this can limit UI features.
- Format configuration option lists as tables, adding defaults, ranges, etc.
- Fix issue introduced in 8.0.20 that prevented network interfaces from showing up in the web UI on non-GM installations.
- The installer for Linux is no longer a static binary in order to run on a wider range of kernels.
- Display each source's status on status tab of web interface.
- Update timekeeper_status.json to include data from all sources.
- Add user-definable names for sources and servers.
- Fixed issue where rsyslog was still using old hostname even after user changed hostname.

v8.0.21 - July 9, 2021

- Validate numeric option values are numeric and if not, send startup trap/syslog message with details.
- Reduce potential for short gaps in PTP source tracking with some GMs when the BMCA requires a clock change in hybrid mode.
- Fixed bug that could cause TimeKeeper crashes when sending management requests (introduced in 8.0.19).
- Fix issue introduced in 8.0.20 that prevented upgrades to TimeKeeper Grandmasters.

v8.0.20 - June 10, 2021

- Add `tkctl` tool for saving/restoring/managing TimeKeeper Grandmaster configurations. See the tkctl documentation for details.
- The configuration for the email relay host and max log age has moved on the Grandmaster web interface, to the “Manage Communication” subtab.
- Fix an issue that prevented basepackage upgrade when version 1.9 or earlier is installed.
- Update packaged Node.js to version 14.16.0.
- Remove support for CentOS/RHEL 5.
- Bug fix to ensure Grandmaster GUI loads after upgrade in some configurations.
- Add HTTPS_KEY_PASSPHRASE to support HTTPS in configurations where the private key is password protected.
- TimeKeeper’s encodesnmptrappassphrase tool has been renamed to encodepassphrase.
- On Windows, retain existing client data and log files when upgrading or reinstalling TimeKeeper.
- Add client search tool to simplify finding client graphs to plot in the TimeKeeper web interface.
- Enable animated Skymap view on TimeKeeper Grandmasters to show signal strength and GNSS visibility over time.
- Fix for plotting clients on the client live graphs in the web interface where the client is only delivering data over NTP (introduced in 8.0.17).
- Added a new section to show how TimeKeeper can work with Telegraf InfluxDB Grafana.
- Bug fix to accept absolute minimum value for SET_TIME_THRESHOLD.
- Fix Windows uninstaller to remove all files when TimeKeeper is running and delete the TimeKeeper service.
- On Linux, synchronize the hardware clock more frequently for out of band systems that rely on it.
- Faster configuration collection and tkstatus execution, especially on Windows. Also avoids potential timeouts displaying details in the web interface configuration tab.
- Correctly optimize hardware timestamp steering when the primary source is on a VLAN interface hosted on a bond.
- Fix memory leak on Windows when exchanging management data.

v8.0.19 - February 25, 2021

- Add SET_TIME_THRESHOLD to ease regulatory compliance, particularly in virtual environments with VM processing delays.
- Fix to allow skymap images to be displayed when using HTTPS.
- Fix to allow timekeeper_cli editing of the timekeeper.conf file on Grandmasters when RADIUS or TACACS+ is enabled.
- Allow more time for web logins for cases where slow or unresponsive RADIUS/TACACS+ servers are in use and local authentication fallback is needed.
- Less time required to startup and more accurate tracking when starting up.
- When a major/minor time source is primary provide a much faster startup that does not require slewing to the minor time first.
- Added a new FILESYNC method for sharing client data for monitoring that can retrieve all old data.
- Add ability to limit times over which client logs are shared.
- Now possible to restrict which client hosts data is shared to limit data transfers.
- Do not allow /31 netmasks (255.255.255.254) when configuring network interfaces on TimeKeeper Grandmasters.
- Prevent startup issues by validating host time and resetting it on startup if it’s earlier than 5 years before the release of TimeKeeper or greater than 30 years after the release.
- Incoming time that is outside of the 5 years before or 30 years after the release of TimeKeeper will be rejected to prevent clearly invalid time.
- Increase the delay before accessing a hot-plug PHC device to 1 second.
- Fix regression that reduced compliance data collected from Windows clients.

v8.0.18 - October 30, 2020

- Reduce GPS offset error under load with VelaSync and Gen 1 TimeKeeper Grandmasters. Issue does not affect Gen 2 TimeKeeper Grandmasters.
- Remove support for 32 bit Linux.
- Increase the number of network card clocks allowed from 16 to 50.
- Fix occasional failed send of announce packet under heavy load.
- Reduced overhead when handling PTP management messages which can reduce GPS sync errors.
- Fix for sourcecheck to prevent failover to inactive source.

v8.0.17 - August 11, 2020

- Fix for false alarms with GPS/GNSS receiver errors on Grandmasters.
- Reduced CPU consumption/overhead.
- Add MINALERTCOUNT to Compliance to only report clients that experienced a specified minimum number of alerts.
- Fix for HTTPS CSR generation that caused invalid attributes to be generated if no attributes were specified (introduced in 8.0.12).
- TimeKeeper web GUI now shows which source is primary over time.
- If the IFACE specified for a source does not exist, the source is not used.
- timekeeper.conf file on unix systems must be readable by non-root users.
- TimeKeeper process listing is now simplified.
- Support privilege-based remote authorization for user access. Please see “RADIUS/TACACS+ support” section about how authentication behavior has changed.
- Fix to allow all interfaces to be available in web GUI on Linux, including those that have not been assigned a network address.
- Solaris service management improvements for shutdown timeout and lockfile handling.
- Fix to avoid reporting stale TimeKeeper status in some cases.

v8.0.16 - May 29, 2020

- Revert so default behavior to send a PTP followup message if non-hardware sync timestamp was found, previously that was not default behavior.
- Allow source failover for more cases where time source data may be active but unusable.
- Changes to make Grandmaster GPS/GNSS tracking more resilient when under heavy load.
- Correct source port for PTP unicast messages. Error introduced in v8.0.12.
- Support PTP Peer-to-Peer delay mechanism (P2P).

- Fix for issue introduced in 8.0.14 that can cause periodic gaps in source tracking.
- Add ENABLE_NIC_BLACKLIST to automatically avoid unstable features known to be present in some drivers.
- Fix snapshot view for Windows GUI.
- Prevent slow memory consumption on systems with slow storage or high I/O load.

v8.0.15 - April 3, 2020

- Reduce overhead of serving telecom profile PTP clients a great deal.
- Fix for GPS/GNSS tracking when grandmasters are under heavy load.
- Provide clearer TimeKeeper service details when Windows is having resource management issues.
- Fix possible GPS/GNSS startup failure on Grandmasters.

v8.0.14 - March 9, 2020

- Improved accuracy of GPS/GNSS time source on TimeKeeper Grandmasters.
- Properly display hostnames that contain periods.
- Prevent spurious GNSS errors being reported on TimeKeeper Grandmaster G2s.
- Improved clock modeling which provides better accuracy.
- If the unicast PTP sync rate is less than 4 at startup, increase it temporarily to speed initial sync.
- Remove support for Windows Server 2008 and Windows 7.
- Reduced TimeKeeper CPU load when interacting with some buggy NIC drivers.
- Further improvement for hardware timestamping support under highly loaded systems.
- Fix for managing interfaces on hosts with high numbers (hundreds) of interfaces.
- Reduce overhead when tracking NTP and PTP client/server statistics under high load.
- Disable arcfour, 3des and blowfish ciphers for ssh access to TimeKeeper Grandmasters.
- Allow PTP traffic without requiring static routes on hosts providing invalid packet details.
- Protect web GUI from terminating Windows processes no longer owned by the web GUI.

v8.0.13 - January 7, 2020

- When user moves source within web-GUI list, have its “UNAPPLIED” label move with it.
- Add optional support for higher accuracy hardware timestamping on Linux.
- Fix web GUI so that the MAJORTIME setting follows the referenced source when re-ordering sources.
- Fix for TimeKeeper Grandmasters to correct for rare log rotation issue under heavy disk usage.
- Fix bug introduced in 8.0.12 where wrong update time was stored in server’s client log for NTP clients.
- Allow more time for hardware timestamp acquisition when under high load, related to feature introduced in 8.0.12 allowing for high PTP packet rates.
- Allow TimeKeeper to associate PHC interfaces to more types of network device names.

v8.0.12 - November 8, 2019

- Permit mixing of HW and SW timestamps on Solarflare cards since some driver/linux kernel combinations cannot receive reliable hardware timestamps all the time.
- Permit whitespace in day-of-week list for compliance reports.
- Accept PTP packets from non-standard ports.
- Greater accuracy through more robust behavior when small numbers of hardware timestamps are missing/unavailable. Some clients may note rare missing PTP followup messages.
- Improved TimeKeeper GM1 and GM2 serving timestamp accuracy.
- Reduced memory and CPU overhead in multicast PTP environments.
- Improved accuracy from better noise rejection when estimating clock frequency.
- Improved accuracy immediately after TimeKeeper starts.
- Fix for skymap feature on time serving appliances, bug was introduced in v8.0.11.
- Increase allowed temperature range for Grandmaster devices to 105F.
- Resolve rare bug where RAID status may not be reported on TimeKeeper Grandmasters, introduced in v8.0.9.
- Only log re-resolving DNS names for NTP sources when the IP address changes or VERBOSE_NTP is set.
- TimeKeeper no longer includes the Symmetricom BC637 and Spectracom TSync drivers, instead it will use versions installed on the host with the SYMMETRICOM_DRIVER_PATH/SPECTRACOM_DRIVER_PATH options.
- Add exclusion of clients from compliance reports based on regex with EXCLUDEDCLIENTSET.
- Support client-requested rates for telecom-profile PTP sync and announce messages. Rates can be specified at the client and are obeyed by the server. (See note for PTPSERVERSYNCRATE.)
- Changed minimum packet rate from 1 packet every 60 seconds to 1 packet every 64 seconds.
- Reduce TimeKeeper’s memory requirements when tracking time sources.
- The HTTPS CSR generation process on TimeKeeper Grandmasters can now specify subject alternative names.
- Allow up to 32 IP addresses per network interface.
- Add official support for SUSE Linux Enterprise Server 15.
- Display hostname in web GUI for clients as well as Grandmasters.

v8.0.11 - June 25, 2019

- Improved accuracy when hardware timestamps are not available for NTP/PTP.
- Improved clock modeling accuracy and noise rejection.
- Ensure TimeKeeper service path is quoted in Windows services.

- Improved delay modeling and rejection of noisy packets/samples.
- TimeKeeper will not reinitialize network interface hardware time stamping if it is already enabled.
- Detect and warn in web interface about network interfaces with addresses on the same subnet.
- Allow empty challenge password when creating a Certificate Signing Request on TimeKeeper Grandmasters.
- Fix regression (introduced in v8.0.10) that disabled SNMP on reboot of GM.
- When license file is replaced/updated it is no longer necessary to restart TimeKeeper if only the expiration date changed. If the features/options have changed in the license TimeKeeper will alert that a restart will be automatically performed when the previous license expires.
- Add official support for Red Hat 8.
- Improved accuracy on Windows Server 2016 and 2019.
- Specifying SERVENTP_IFACE no longer affects NTP sources and their intended interface.
- Fix communication issue when TimeKeeper is started without access to DNS resolution and using hostnames in source definitions.
- Fix for Linux kernel bug on some Skylake processors that resulted in bad clock frequency which resulted in constantly increasing offsets.
- When hot-plug PHC devices are added pause before using them to work around race conditions in some PHC drivers.

v8.0.10 - April 15, 2019

- TimeKeeper will continue startup even if DNS resolution of a configured PTP/NTP server fails.
- Disable unused services on TimeKeeper Grandmasters.
- Compliance can now report on issues only related to a client's primary source with the ONLYREPORTPRIMARY option.
- Avoid an error with SERVENTP_IFACE when used on a bonded interface that also has IP aliases assigned.
- Fix regression (introduced in v8.0.9) that prevented Compliance report numbers > 50 from being used.
- Fix memory leak when periodically building set of clients and sources for Compliance to analyze.
- Add hostname and date to problem report name to aid in identification and filename uniqueness.
- Add support to optionally output tkstatus information without any truncation. When tkstatus is output without truncation, one-way delay values are also displayed.
- Fix NTP server performance regression, did not affect accuracy but just queries/second.
- Provide more hardware timestamping support for interfaces with custom controls and VLANs not related to the underlying device name.
- Faster startup when primary time source is unavailable and a secondary source must be used.
- Alert when an impossible or likely erroneous UTC/TAI offset change is seen from a PTP Grandmaster.
- Add support for NTP authentication on Windows.
- If the host time is detected too far in the past, set it to known good value before starting TimeKeeper licensing checks.
- Fix certain out of disk situations that could cause a partial factory reset on Grandmasters.

v8.0.9 - February 22, 2019

- Windows installer and uninstaller improvements.
- Accuracy improvements due to improvements in frequency prediction.
- Correct previous fix PPS output for limited number of TimeKeeper Enterprise Gen 1 Grandmasters that had reversed polarity output PPS. Fixes PPS output error on G1 Grandmasters introduced in v8.0.7.
- TimeKeeper Enterprise Gen 2 Grandmasters now include cable delay in output PPS.
- Save latest TimeKeeper start-up information for improved support. The file timekeeper_init or timekeeper_init.log for Windows is saved in the log directory when TimeKeeper starts timing updates.
- Fix for sending SNMP traps from TimeKeeper on Windows 2012 and earlier.
- Fix hardware timestamps for Solarflare cards using bonding/VLANs on older (pre hardware timestamp) Linux kernels.
- Add support for hardware timestamping on Linux teaming interfaces.
- GLONASS satellite system is no longer enabled by default.
- Enable SBAS and QZSS systems when GPS is configured on TimeKeeper Enterprise Gen 2 Grandmaster.
- Support hardware timestamps on VLANs even in cases where kernel-provided details are incomplete.
- Compliance now delivers PDFs directly in the TimeKeeper web interface for easier handling of audit data.
- Compliance performance improvements in delivering access to audits and more detailed Excel exports.
- Provide clearer summaries of client warning and alert details in Compliance audits.
- Don't let corrupted network-configuration file on Grandmaster prevent network configuration from showing up in web UI.
- Allow problem reports to be generated in user specified directory.
- Fix to prevent extra logging when hot-plug network device events are interrupted.
- Add official support for Windows Server 2019.

v8.0.8 - November 20, 2018

- Automatically disable remote user authentication when all servers are deleted, prevent it from being enabled if no servers are defined, and disable the other (RADIUS or TACACS+) when one is enabled.

v8.0.7 - November 2, 2018

- Add support for hot-plug network devices so that their clocks are steered properly.
- Fixes for old Solarflare API (RHEL 5 and similar systems).
- Fix to not report "Unknown" oscillator type on GrandMasters when GPS/Oscillator is not primary/active time source.
- Fix PPS output for limited number of TimeKeeper Enterprise Gen 1 GrandMasters that had reversed polarity output PPS.
- Fix for possibility of spurious (momentary) reporting of TimeKeeper Enterprise Gen 2 GrandMasters in holdover when they are not.
- Have timekeeper_cli change hostname immediately rather than upon reboot.

- LOWCPU mode will be enabled automatically for high-noise time sources that cause excessive CPU use.
- Improved holdover performance through better clock modeling.
- Be more restrictive during shutdown to only kill TimeKeeper's web service processes.
- In addition to "M", recognize "MD5" as representing the MD5 algorithm in the NTP key file.
- Add support for sending SNMP traps from TimeKeeper on Windows, in addition to reporting events in the Windows event log.
- Support "vsyscall=none" kernel parameter on Linux systems.
- Remove Save configuration and Restore configuration from Grandmasters.
- No longer force W32Time to use NoSync, instead warn if invalid configurations are detected.
- Provide additional HTTPS headers for enhanced web security.
- Fixes to record management/monitoring responses from different types of clients.
- Updated SNMP MIB revision and last-updated fields.
- Removed automated one-click upgrade on GrandMasters due to policy changes. Upgrades of TimeKeeper and the basepackage are done via the TimeKeeper GUI as before.
- Fix GUI discrepancy where PTP servers with management queries/responses disabled would display the options as enabled.
- Add support for relay hostname with hyphens when entered in the Grandmaster's web GUI.
- Compliance allows custom start and end times and reporting only for specific days of the week.
- Fix regression (introduced in v8.0.0) so that TimeKeeper includes hostname in Subject header of email alerts.
- Grandmasters now properly alert when inlet temperature goes out of range.
- Include more detailed GPS state on Grandmasters and fix possibility of not reporting holdover state on status page.
- Fix regression (introduced in v8.0.0) so that TimeKeeper on Windows once again uses the specified IFACE.
- Fix regression (introduced in v8.0.0) so that user can update the HTTPS certificate on a Grandmaster.

v8.0.6 - August 26, 2018

- Fix occasional lost packets due to Linux errors that mis-identify network interfaces. This manifested as lost PTP sync for brief periods of time.
- Request more time from Solaris Service Management Facility to avoid TimeKeeper being put into maintenance mode in some situations.
- Fix for hardware timestamps on bonded VLANs with Solarflare cards.
- Remove requirement for TSC as the only clocksource.
- Fix memory leak in clock modeling. This was introduced in v8.0.3.
- Prevent tkstatus.bat error messages when run as non-admin user on Windows.
- Fix Compliance issue that prevented clients from having all of their sources included in audits.
- Fix the GUI on TimeKeeper Grandmasters when deleting bonds and VLANs.

v8.0.5 - August 2, 2018

- Fix for PTP servers and clients when asked to relay local timing client data to remote hosts.

v8.0.4 - July 30, 2018

- report_problem.sh, when instructed to upload the problem report, will attempt HTTPS first then fall back to FTP if HTTPS upload fails.
- Fix occasional failed send of packets on Grandmaster hardware.
- Fix for Grandmaster 10/25G interfaces with VLANs on bonds. Previously this would revert to incorrect active-active rather than the correct active-passive state. Now proper active-passive is configured.
- Improved outlier rejection. During periods of rare but large network noise/jitter, TimeKeeper will better reject inaccurate data and maintain time more accurately until network noise decreases.
- Fix PTP telecom profile configurations where the wrong network interface was selected by TimeKeeper when it was not the default route.
- Updated Grandmaster antenna location map to work with recent Google maps API changes.
- Improved accuracy from better filtering that results in more precise frequency estimate.
- Fix labeling of Compliance reports via web interface from GMT to UTC.
- Fix for VLAN hardware timestamps and specific fix for Solarflare VLAN and bonded interfaces.
- Fix for hardware timestamp cards that can only timestamp PTP - Solarflare and Broadcom among others.
- Make /var/log/messages readable by admin on Grandmasters.
- When displaying weekly/monthly/yearly Compliance audits, provide more clearly empty daily audit summary for cases where there are no active clients/sources for that day.
- Fix numbering for the sourceQuality and sourceState SNMP traps so they're more easily matched to the source with the error.
- Bug fix for possible crash when exchanging management data.
- Fix for PTP servers delivering unicast traffic in some non-IPv6 environments.

v8.0.3 - June 25, 2018

- TimeKeeper now catches installation errors on startup by validating the install and will not start if installed files have been modified.
- TimeKeeper will no longer respond to its own PTP GET requests. This avoids unnecessary network traffic.
- Add support for Grandmasters to send email using an external SMTP relay.
- Compliance reporting speed improvement due to many optimizations - in some cases a 15x improvement in report generation time.
- Faster Compliance startup. Previously generating lists of clients on startup consumed a great deal of time with 10,000+ clients. Now it is much faster (on the order of seconds).
- LOWCPU option to prioritize low CPU consumption for a given source. This will reduce accuracy and outlier rejection to some degree but greatly reduce CPU consumption.
- Improved accuracy and reduced CPU consumption for higher sync rates (higher than 5 updates per second) on NTP/PTP sources.
- Reduce the number of packets sent over the network when hardware timestamps are available since 'pre-cache' packets are not necessary.
- Problem reports can now be launched from timekeeper_cli on grandmasters.
- Added alerting on Grandmasters for early warnings about possible disk failures.

- Removed size restrictions on previous Compliance audits when converting data to SQL database.

v8.0.2 - June 4, 2018

- Compliance generates reports faster due to improvements in PDF and HTML generation. In some cases a 4x reduction in time needed per report.
- On startup log a message when a global parameter is specified for a specific server or source to indicate that it is being ignored and is incorrectly configured.
- Previously TimeKeeper would query network cards listed in AVOID_IFACES for very basic information about them. This release no longer does that to reduce interaction with AVOID_IFACES network cards.
- Add option (PTP_OVERRIDE_BC_PRIORITY) for PTP servers that allow overriding the upstream PTP priority 1 and 2 fields with configured fields when in boundary clock mode.
- Allow TimeKeeper Grandmaster to be queried for system parameters via SNMP (see the "SNMP MIB" section for details).
- Early detection (up to several months) and reporting of a leap second pending when advertised by GNSS systems for generation 2 TimeKeeper Grandmasters.
- Allow deletion of VLANs from the web interface of TimeKeeper Grandmaster devices.
- Fix issue preventing recent syslog content from being viewable in the web GUI on TimeKeeper Grandmasters.
- Allow Grandmaster admin user to run tcpdump.
- Allow ENABLE_MANAGEMENT_QUERY and ENABLE_MANAGEMENT_RESPONSE to be set on a per PTP source/server basis for finer control on which interfaces and PTP sources/servers exchange management data.
- Fix single-OID SNMP query of per-source data so that it agrees with the results of a walk. Previously, the leaves corresponded to source numbers (minus one); now, the leaves are consecutively numbered and one must inspect the sourceNumber.
- Added GNSS and timestamp-type SNMP OIDs for the current source and on a per-source basis.
- Fix web UI so that it allows -1 for the web management port (this disables HTTP access while still allowing HTTPS).
- TimeKeeper will detect and stop other time sync daemons when TimeKeeper is configured to control the clock. It will not permanently disable any other time sync services but will stop running programs.
- Add configuration option ENABLE_COMPLIANCE_DNS to switch on or off DNS resolution for compliance audits. Improves performance with DNS off for compliance reports.
- Grandmaster system image save/restore now does not save/restore the device's serial number.
- Grandmaster will now alert when it fails internal health checks. These alerts will be logged wherever configured. See the "grandMasterFaultTrap" section for details.
- Make problem report collection easier when the system running TimeKeeper has an invalid network configuration.
- Handle longer CLIENTSET values better.
- Fix issue where specified TTL setting may not be applied to non-IPv6 PTP sockets.
- Compliance generates reports more frequently for intraday checks as well as faster configuration, startup, test cycles and deployment.
- IPv6 support is required only if an IPv6 source or server is configured.

v8.0.1 - April 17, 2018

- Faster start of Windows 2008 in some cases. Windows 2008 must send NTP queries slower and cannot startup as quickly so do not attempt to use too many samples which delays startup.
- Compliance tab will now show estimated time until all audits are complete and show how long processing of a single daily audit requires.
- Fix 'stripdata' tool for reducing size of stored data files to handle additional column of data that was recently added.
- Compliance now includes the report number in addition to title in the header of generated audits.
- Allow user to request sync error threshold alerts to be sent only for the primary source
- Reduced number of sockets for PTP and NTP sources. This reduced overhead a great deal on grandmasters and systems serving time.
- TimeKeeper will now detect when it is instructed to use a NIC that does not have a valid IP address and report the error.
- TimeKeeper, on a TimeKeeper Grandmaster, more tightly controls which interface it replies to requests on. This ensures extra replies are not sent on incorrect interfaces which avoids unnecessary and incomplete ARP requests.
- Remove PTP logmessageinterval workaround in announce messages. Previously TimeKeeper would advertise a lower message rate than it actually used to support some buggy PTP clients that timed out earlier than they should. Now TimeKeeper correctly advertises how often it sends announce messages.
- Improve behavior with higher sync rates by not waiting for hardware timestamps or software timestamps for transmit when they won't be available (no hardware timestamp hardware available and software TX timestamps when using unicast messages).
- Allow user to request sync error threshold alerts to be sent only for the primary source.
- Added Excel file export facility from compliance_query for compliance data.
- Windows service will be stopped if the TimeKeeper application stops for 5 or more minutes.
- Force the displayed total time spent in a warning/alert state to 0.0 when building weekly/monthly/yearly Compliance audits that have no clients to report in the audit window.
- Compliance can now capture and report on upstream accuracy from NTP-only non-TimeKeeper clients.
- Fix Compliance issue with capturing alerts at the end of the UTC day.

v8.0.0 - March 16, 2018

- Upgrades, speedup, improvements to TimeKeeper Compliance. SQL database access to reports, much faster report generation, more reliable operation, faster startup and initial reports. See Compliance documentation for details
- TimeKeeper web GUI now supports HTTPS even when not on a TimeKeeper Grandmaster, see web configuration for details
- Add IPv6 support on Linux for software-only installs of TimeKeeper (not on grandmasters)
- Logging of the sum of all time error between a client and multiple upstream servers for recording of "time error budget". This is the sum of accuracy for each time source upstream until true UTC source is reached. See the documentation for more information.
- Previously some upgrade paths on grandmasters did not update all the NIC drivers. This release corrects that so NIC drivers are always properly updated to the latest version.
- SERVENTP_IFACE still binds to device, but socket will not show as bound to specific IP address in netstat.

TimeKeeper Version 7

TimeKeeper Version 7.2.x

v7.2.18 - April 16, 2018

- Bug fix for possible crash when using ENABLE_DETECT_ASYMMETRY
- Prevent possible crash when processing invalid PTP management data
- Prevent crash when trying to manage Solarflare UUID filtering on systems with large routing tables

v7.2.17 - March 5, 2018

- Fix for static routes on bonded interfaces on grandmasters. Previously static routes could be lost/ignored on startup on bonded interfaces, fixed in this release.
- Fix asymmetry detection/reporting, previously would not report correct values
- Make log rotation behave more intuitively than what the logrotate utility provides. When changing 'maxage' to a smaller number logrotate will leave old files in place. Now they'll be properly deleted as one would expect.
- Add optional alert for sources that are non-primary when they stop responding for more than 3 minutes
- Fix cabledelay parameter by correcting inverted logic (sign was previously reversed)
- GrandMaster serial number printed at login prompt for easy reference
- Updates to take advantage of latest basepackage (v1.12) and updated Spectre/Meltdown fixes
- Add support to allow the logging directory to be changed on Windows

v7.2.16 - February 14, 2018

- Changes to handle accuracy limitations/packet rate on Windows 2008
- Accuracy improvements in the presence of periodic jitter
- Removed limit on size of kernel routing table
- Add support for binding to a network interface on Windows
- On Windows, support all interface types, not just "Ethernet," and ignore those without an IP address
- Better memory management for Compliance

v7.2.15 - January 19, 2018

- Allow transparent clock correction in sync or followup packets

v7.2.14 - January 12, 2018

- General security improvements to support CVE fixes on grandmasters
- Grandmaster factory reset will now reset the IPMI password to factory setting
- Reduced CPU use on SPARC
- Skip invalid packets/packet data when calculating NTP server egress delay
- Added WINROTATECOUNT option to set the number of TimeKeeper logs retained on Windows
- EULA updated
- Added source IDs to changedSourceTrap text
- Support VERBOSE_TCPDUMP on Windows
- Added source traceability information to timing log files
- Compliance/monitoring only reports configured sources via management query.
- Add license enablement for TimeKeeper on VelaSync Grandmasters, see basepackage release notes for details

v7.2.13 - November 27, 2017

- Fix Solarflare hardware timestamps for non-hardware timestamp Linux kernels
- Avoid truncated input in timekeeper_cli
- Bug fix for possible crash on Windows
- Update Microsemi BC635/BC637 driver for Linux
- Added DONOTROTATELOGSONSTARTUP option to prevent rotating logs when TimeKeeper starts up
- Changes to Linux logrotate config so older files are properly deleted, see section on service management for details
- Do not log NTP client data from clients that provide invalid data
- Improved support for 2nd generation TimeKeeper Enterprise Grandmaster
- Increase size of ARP table on grandmaster appliances
- Fixes for TAI/UTC offset when steering a NIC directly to a PTP source rather than host clock
- Improved accuracy for high rate PTP/NTP sources

v7.2.12 - October 18, 2017

- Reduce CPU overhead when not steering system clock
- Solaris/SPARC support no longer beta
- Updated documentation format
- Solaris/SPARC bug fix related to ptp management messages
- Fix PPS output on FSMLabs NIC for some configurations

v7.2.11 - October 2, 2017

- Optimization so PTP management responses are only send on interfaces they are received on
- Fixes for grandmasters that are under heavy disk/CPU load

- Solaris uninstaller and init script fixes
- Solaris 10/sparc 64-bit support (beta)
- Fix installer for Linux 32-bit systems
- Add *raw* one-way delay to existing one-way delay graph in web GUI
- Report *Oscillator Type* in the status report for grandmasters in web GUI
- Achieve initial sync faster on Windows 2008R2
- Only use well-known ports
- Add button to web GUI that sends a test alert
- Stop TimeKeeper on Windows 2016 faster
- Increase NTP response timeout to 750ms

v7.2.10 - August 7, 2017

- Solaris bug fix for clocks that need very large correction
- Improve accuracy of initial frequency and offset modeling at startup
- Enable global and per-server ENABLE_HWTSTAMPS configuration parameter
- Add ALLOW_UNREASONABLE_UTC configuration parameter to accept PTP sync messages whose offset is not known to be correct

v7.2.9 - July 26, 2017

- Allow silent install option to apply on Windows when an elevated context is used
- Fix bug with running timekeeper_cli on TimeKeeper Grandmasters (regression in 7.2.8)
- Remove time adjustments made by other applications on startup for faster TimeKeeper sync
- Prevent Solaris from interfering with TimeKeeper steering the time
- Reduce amount of PTP lease logging

v7.2.8 - July 24, 2017

- Improved clock modeling on Solaris
- Fix issue with collecting problem report data from the web on Windows (regression in 7.2.7)
- Add support for svcadm on Solaris
- Fix case of PTP source without an explicit IFACE setting not receiving intended PTP data
- Ensure PTP sync/announce messages use sequential sequence IDs when unicast clients are present

v7.2.7 - June 22, 2017

- Update for newer Velasync Rubidium units
- Fix for possibly slow startup on Velasync Rubidium units (regression in v7.2.6)
- Fix NTPSYNCERRORTHRESHOLD setting
- Fix and performance improvement for handling high management message volumes
- Support for Solaris 10/11 for 64-bit x86
- Bug fix of possible crash and performance improvement for handling high management message volumes
- Improved accuracy when steering NIC clock to remote time source directly
- TimeKeeper no longer needs gcc on install
- End support for RedHat 4, Centos 4 and older distributions
- Fix packaging issue on SLES 11 where too many files are included in the timekeeper-config RPM

v7.2.6 - June 7, 2017

- Fix for grandmasters to ensure consistent interface names
- Update to latest Spectracom Tsync driver
- Argument to installer “-s” will prevent installer on windows from requiring shell interaction
- More accurate hardware timestamps
- Improved accuracy and improved rejection of noise
- MONITORONLY source data is not exchanged via PTP management queries
- Fix for possible crash with many PTP clients/servers and high management message volume
- Add support for “tkstatus” with tkstatus.bat on Windows
- Fix for sources tracking PTPv1 grandmasters distributing non-UTC based time
- Fix command-line network config to, like web GUI, prevent invalid settings

v7.2.5 - April 23, 2017

- Fix case where sourcecheck might not detect all problems introduced in v7.2.3
- Fix case where multiple PTP sources/servers may not respond to management queries
- Support for more PPSDEV and PPSSOURCE combinations
- Support for u-blox devices and multiple GNSSes, such as Galileo and GLONASS
- Fix off-by-one error in management data exchange
- Support for /dev/ttyACM* devices
- All sockets default TOS to “low delay” now
- Restore serial number/base package version display on grandmaster hardware
- Fix for RHEL 4/5 system frequency bugs for faster TimeKeeper startup

- Fix occasional errors with updating temp files on Windows
- Package web authentication files with the timekeeper-config RPM
- Add official support for Windows Server 2016
- Handle management delivery of client data when very slow query rates are used
- Fix cases where PHC device could be used for an interface listed in AVOID_IFACES

v7.2.4 - March 6, 2017

- Work-around Linux kernel bug that incorrectly sets the commanded clock frequency
- Fix for possible false reporting of failed power supply on grandmaster appliances
- Fix RPM install so that system startup files are updated properly
- Resolve issue that stopped logging of NTP client data if we also receive PTP data about that client
- Fixes for Velasync/TimeKeeper Enterprise Grandmaster power supply failure detection
- Support for updated Velasync hardware
- Add support for SNMPv3 alerting on TimeKeeper clients
- Fix for Windows Event Source not being registered
- Add configurable ALERTTHRESHOLD and WARNINGTHRESHOLD for Compliance
- Disabled reverse path filtering on VLAN and bonded interface on TimeKeeper Grandmasters
- Allow PHC devices to be steered directly to upstream sources in additional cases
- Prevent sync error alerts when initial sync may not be complete

v7.2.3 - January 17, 2017

- Fixes for report problem script when disk space is limited
- Fixes for Solarflare specific ioctl when PHC unavailable for VLAN/bond interfaces
- service timekeeper status will now return error code when timekeeper is stopped
- Add support for Mellanox Infiniband hardware timestamps
- Document installation/operation requirements
- Fix for passwords that include %%
- Prevent umask from limiting TimeKeeper installation permissions
- Fix when upgrading grandmasters from very old installs (TimeKeeper 6.3.15 and prior)
- Allow disabled sync error threshold throttling with SYNC_ERROR_THRESHOLD_THROTTLE=0
- Add Description field for GM network interfaces
- Send syslog/SNMP trap/email message when bond active slave changes
- Documented required packages for TimeKeeper Compliance - font packages are required
- Avoid memory leak on Windows
- Fix for Windows service not restarting when controlled from the Windows Management Console

v7.2.2 - November 29, 2016

- Documentation updates
- Fix minimum TLS version supported configuration option
- Fix SNMP trap bug introduced in 7.2.1

v7.2.1 - November 22, 2016

- Detect NTP marker that NIST uses for unhealthy servers
- Drop support for obsolete Solarflare and Broadcom timestamp API
- Improved Solarflare support for Linux kernels lacking PHC support
- Allow more characters in SNMP community string
- Fix for possible crash on Ubuntu 16
- Fix for ‘check for new version’ on GUI
- Fix so old files will always be removed when rotated
- Register with windows as installed and register uninstaller
- Fix for network device re-ordering on Velasync/FSMLabs Enterprise Grandmaster
- Fixes for Linux PPS API support
- Default to sending NTP followup messages for improved accuracy when serving NTP
- Allow user to disable older TLS/SSL transports and ciphers for web GUI
- Provide better configuration error detection, improve configuration documentation
- Detect and alert more cases of other applications interfering with clocks on Linux

v7.2.0 - October 11, 2016

- Add support for ARM processors
- Add support for Linux PPS API
- Add support for Linux GPIO API for PPS input
- If PTP/NTP leap second indicator/warnings are received, propagate them to clients
- Improve local holdover oscillator disciplining from PTP/NTP
- Add support for multiple FSMLabs PPS input cards

- Add per-source ability to disable hardware timestamps
- Improve asymmetry detection logging
- Performance improvements for PTP management messages (far less overhead)
- Accuracy improvements for very low sync-rates
- Add NTP MD5 authentication support
- Alert when primary PTP source changes PTP reported clock class or time source
- Improved support for generic NMEA devices - GPS reception mapping
- Fix to correct projection on GPS reception sky map
- Like PTP, NTP servers can trigger alerts if any of the clients report a sync quality that exceeds NTPSYNCERRORTHRESHOLD
- Prevent TimeKeeper from allowing sources and servers on a bond interface member to avoid bond flapping
- On TimeKeeper Grandmasters, allow SSH to be enabled or disabled on specific network addresses
- Add TimeKeeper Compliance
- Send SNMP traps for system clock being driven by something other than TimeKeeper
- Always display GPS tab in web GUI if GPS generating data, not just for GM
- Add to web GUI per GPS, a Signal Strength tab and, on Antenna Locations map, a marker
- Add support for animation of skymap over time to GPS tab
- Update bundled Spectracom Tsync driver to 3.20
- SNMP MIB discrepancy fix - do not require a trailing 0 when querying the number of sources
- Add MONITORONLY configuration parameter
- Handle noncompliant NTP server (incorrect reference identifier)
- Steer Solarflare oscillators directly as before but in more system configurations
- Fix for data consistency issue under certain packet loads
- Detect and handle invalid PTPv1 UTC offset information
- Detect and handle occasional vendor-specific NIC API failures

TimeKeeper Version 7.1.x

v7.1.9

- Handle noncompliant NTP server (incorrect reference identifier)
- Steer Solarflare oscillators directly as before but in more system configurations
- Fix for data consistency issue under certain packet loads
- Detect and handle invalid PTPv1 UTC offset information
- Detect and handle occasional vendor-specific NIC API failures

v7.1.8

- Improved leap second performance when server is slewing
- To make sure the network is fully ready, systemd integration depends on network-online.target
- Correct reported header lengths for some types of PTP management packets

v7.1.7

- Fix asymmetry detection
- Add support for automatic steering and control of custom data processing NIC
- Handle case where Linux assigns high indexes to virtual network interfaces
- Fix case where two concurrent service start commands could run two instances of TimeKeeper
- Catch cases when some gps devices lose utc offset but don’t report it
- Fix log rotation of tcpdump files so they are not corrupted
- Alert when primary PTP source changes PTP reported clock class or time source
- Allow TTL for PTP sources to be editable in GUI
- Detect and support hardware timestamping on interfaces named like VLAN devices

v7.1.6

- Fix report_problem.sh for cases where the log directory is a link
- Improved accuracy for MS Windows systems
- Better integration with systemd-based Linux distributions
- TimeKeeper SNMP tree walks are now much faster
- Users can now edit timekeeper.conf from timekeeper_cli on TimeKeeper Grandmasters
- Sync error detection now alerts even if sources are so noisy they never achieve an initial sync
- Updated TimeKeeper Grandmaster/Velasync basepackage to 1.7 (for CVE-2015-7547)
- Allow more types of GPS-backed sources in the PPSDEV section in the TimeKeeper web GUI
- Improvements to automatic “health check” tools on TimeKeeper Grandmasters
- Fix for web GUI to allow configuration of PPS-based sources on specific network interfaces
- Documentation updates
- Force bonded interfaces on TimeKeeper Grandmasters to always be in active-passive mode
- Fix for TimeKeeper Grandmasters where VLANs on bonded interfaces can be removed unintentionally

v7.1.5

- Fix for polling PPS reads when system time is in the future on startup
- Improve performance for some low quality time sources
- Handle bad/invalid startup data from some GPS models
- Fix for some more than one PPS input configurations
- Fix for product/version/vendor info on map in some setups
- Fix problem in graphs of 'client' timing quality that occurred in some situations
- Fix issue with missing problem report data on Windows systems
- Fix for possible lost messages in log file on startup
- On install, any existing logrotate files will be left as they were on Linux
- "timekeeper_cli" allows per-interface routes and rules on bonded interfaces
- In Windows GUI, allow users to bind PTP servers and sources to network interfaces with spaces in their name
- On Windows, avoid packet processing delays and errors when handling many PTP packets.
- Fix for license expiry details reported via SNMP and with "tkstatus"
- Fix configuration GUI for PTP Sync Error Threshold.

v7.1.4

- HTTPS certificates on TimeKeeper Grandmasters are now signed using SHA256
- Fix configuration save issue when enabling "sourcecheck" feature on web GUI
- Fix configuration save issue when with some source and VLAN/bond configurations on web GUI
- Avoid false error in web GUI when user has configured MAJORTIME option on sources
- Status page uses satellites in 'use' not 'view' to avoid confusion
- More detailed jam indication reporting on FSM Enterprise Grandmaster/Velasync
- Improvements for 'lowquality' mode
- Faster correction of frequency on startup
- Add additional support for hardware timestamping on bonded VLAN interfaces

v7.1.3

- Specify gzip in logrotation for distributions that use other compression
- Fix for tkstatus and service timekeeper status showing wrong status occasionally
- Do faster queries for longer on startup for NTP sources to speed up initial sync
- Added more configurable options for PTP unicast clients
- Fix for startup condition that could cause lack of GPS sync
- SNMP community string is now configurable on TimeKeeper Grandmasters
- Fix for Symmetricom BC637 when GPS module not available
- Fix for older Intel NIC firmware that does not publish PPS events
- Fix for startup condition that could cause non-working output PPS on NetCard
- Set PTP traceable parameter in announce messages
- Performance improvement for tkcat
- Fix for tkcat regression that prevent it from reading older/compressed data
- Support for bonded interfaces on TimeKeeper Grandmasters
- Fixes to configuration interface on web GUI
- Support authentication with multiple TACACS+ servers
- Add support for a list of administrator users in addition to "admin" on TimeKeeper Grandmasters
- Improved timing graph performance
- Improved timing map performance, added update age control on map
- Faster service startup times when archiving existing TimeKeeper client data
- Improve PTP/NTP serving capacity
- Faster restart of TimeKeeper when rotating several log files

v7.1.2

- Fix for possibly NaN results in log files with very high accuracy sources
- Improvements to "self" time source for smoother local oscillator tracking
- Fix for frequency cross-check with "sourcecheck" feature
- Fix for TLV length field in some management messages that some PTP clients report as errors
- Improved holdover accuracy when using NTP/PTP as primary time source
- Updated TimeKeeper Enterprise Grandmaster/Velasync basepackage to 1.6 (new kernel, new drivers)
- Increase serving capacity of TimeKeeper Enterprise Grandmaster/Velasync combined with basepackage 1.6
- Fix possible livelock when hardware timestamps were explicitly disabled
- Fix possible reversed polarity on PPS output of FSMLabs NetCard
- Fix possibility of failed startup due to bad TSC rate
- Fix a potential case where GPS state could be incorrectly indicated very rarely on Velasync
- Faster initial sync of holdover oscillator on TimeKeeper Grandmaster/Velasync

- TimeKeeper can now store logs and data in a configurable location, not just /var/log on Linux
- Fix Windows documentation install issue
- Make problem report location more clear for Windows users
- Avoid case where “tkstatus” and “service timekeeper status” did not provide same output
- Work around PTP unicast lease handling restrictions with some Juniper routers

v7.1.1

- Improve NTP accuracy
- Fix possible crash on Windows with certain network devices
- Web timing graphs visibly disable a source when there’s no data for the selected time window
- Fix WEB_MANAGEMENT_IP bug that caused it to not work properly
- Fix graphing issue with some NTP servers that reply with spaces in their data
- Display Hardware Vendor on timing map for PTP nodes

v7.1.0

- Fix to allow disabling hardware timestamps again
- Add beta MIPS64 support
- Add Windows support
- Improvements in accuracy for all sources
- PTP client/server can now use syncrate up to 128 per second
- Support hardware timestamping on bonded interfaces
- GUI performance improvement when many sources are active
- PTP serving correction to properly advertise when in holdover state.
- Velasync/TimeKeeper Grandmaster output PPS will now match system time when operating in non-GPS mode
- Updates and Improvements to Timing Map

TimeKeeper Version 7.0.x

v7.0.7

- Fix status when tracking a Spectracom TSync card with 0 satellites
- Disable reverse-path filtering to allow PTP serving on misconfigured networks

v7.0.6

- Accuracy improvements for NTP/PTP client mode
- Improve accuracy of NIC oscillator steering

v7.0.5

- Fix for TimeKeeper segv on startup when initialization fails
- Add option to steer appliance oscillator from non-GPS sources
- Fix for appliances that could show periodic outages without proper DNS setup
- Improvements to oscillator steering on Velasync/TimeKeeper Grandmaster
- Ensure IPMI time is correct on TimeKeeper Grandmaster systems
- Fix for ‘status’ tab that could cause it to be blank occasionally
- Allow IPMI network gateway to be set for TimeKeeper/Velasync appliances
- Fixes to detect and reject malformed PTP data
- Documentation updates
- Avoid sending license alarms until they’re necessary
- Detect and reject invalid satellite details when building signal strength map
- More detailed information when alerting about RAID device failures on TimeKeeper Grandmasters
- Fix PHC steering in the case where the NIC is steered directly to the upstream source and that source is lost
- Fix regression in network timing map where nodes may not be connected accurately
- TimeKeeper Grandmaster allows restoration of original /etc/timekeeper.conf via “timekeeper_cli” tool

v7.0.4

- Fix for TAI timescale PTP client leap second bug
- Handle case where multiple connect/disconnect cycles of antenna can cause unstable GPS frequency
- Add GPS tab to appliance to display signal strength map and location map
- Fix problem with RS232 PPS input on 32-bit systems
- Do not emit traps/alarms when switching sources during startup, only during normal operation
- Always force timezone to UTC on TimeKeeper Grandmasters for logging consistency
- Fix for installer for case where an invalid license file is provided
- Fix web GUI issue when saving recently added options
- Fix issue where very long SNMP traps/emails/syslog messages could be truncated

v7.0.3

- Improve accuracy/tracking of low-quality time sources with adaptive smoothing
- Remove cosmetic problem of message in logs of related to *sfc_read_skew*
- Fix regression (from 7.0 release) that meant very low quality sources were not handled as well as they could be
- Fix for NTP specification ambiguity so root dispersion/delay are handled properly no matter the NTP client/version
- Accuracy improvements with all source types when system frequency is less stable
- Fix bug introduced in v7.0.0 for Tsync cards that have no GPS module (only PPS)
- Make VERBOSE_NTP configurable via the web interface
- Add ENABLE_SATELLITEDATA option and collect detailed GPS satellite angle/signal data
- Web interface specifies default PTP priority values in PTP server configuration
- TimeKeeper Grandmaster now reports GPS state on status page even when using another source
- TimeKeeper Grandmaster provides a simpler 'GPS source' selection
- TimeKeeper Grandmaster also now allows user to configure gateway and reboot appliance from the command line with timekeeper_cli

v7.0.2

- Fix chassis state information on Velasync
- Correction to frequency graph axis label - ticks/second is now properly labeled as PPB change
- Avoid opening any PHC devices related to an interface named in AVOID_IFACES
- Disable per-device gateway settings on TimeKeeper Grandmasters, replaced with advanced routing capabilities
- TimeKeeper Grandmaster supports configuration of policy based routing via timekeeper_cli

v7.0.1

- Update to 10G grandmaster network drivers to support new hardware
- More improvements in NTP accuracy when hardware timestamps are available
- Do not cause unnecessary ioctl messages from Solarflare cards
- Prevent potential web interface restarts under heavy load
- Raise process limits on TimeKeeper Grandmaster
- Control time on Solarflare cards even when Linux API is not available (useful for SolarCapture and PPS output)

v7.0.0

- Grandmaster configuration of IPMI GATEWAY works correctly now
- Expand holdover support for devices that can support it (Spectracom TSync and others)
- "tkstatus" command does not show PHC/NIC time devices (sources 100+)
- Fixes for "service timekeeper status"
- Greatly reduce overhead of PTP/NTP serving which also increases client capacity of grandmasters
- Reduced memory footprint
- Add ability to detect/report asymmetric network links
- TimeKeeper threads now show their function in certain "ps" commands rather than "timekeeperapp"
- Support for Spectracom SecureSync
- TimeKeeper grandmaster bug fix that could interrupt service for a short time when 0 satellites were visible
- Fix issue preventing web client graphing involving clients with fully qualified domain names
- Fixes for non-primary PPS input on some NIC cards
- When tracking and serving a given PTP domain TimeKeeper will not track itself
- TimeKeeper install no longer requires kernel module or kernel development package
- RAID support on TimeKeeper grandmaster
- EMAILNOTIFICATION_THROTTLE allows for the optional throttling and bundling of email message alerts
- INITIAL_SERVE_ACCURACY optionally prevents serving of time before a named accuracy threshold is met
- Support for hybrid mode PTP with Grandmasters from Oscilloquartz
- Ability to save and restore appliance configuration state on TimeKeeper grandmasters
- timekeeper_cli command line tool to configure some TimeKeeper Grandmaster capabilities from the command line
- Basepackage upgrade (version 1.3) for TimeKeeper Grandmasters available to address GHOST glibc vulnerability (CVE-2015-0235)

TimeKeeper Version 6.3.x

v6.3.19

- Fix for TAI timescale PTP client leap second bug

v6.3.18

- Faster leap second correction in some situations

v6.3.17

- Reduce unneeded logging when not in verbose modes
- Handle invalid DNS lookups when reresolving hostnames at runtime
- TimeKeeper init script now checks for both kernel and user components when checking service status

v6.3.16

- clock_gettime(CLOCK_MONOTONIC) no longer pauses briefly when clock is adjusted (CLOCK_MONOTONIC was always monotonic before, however)
- Fix for bug that could allow time on some NICs to oscillate
- Fix issue preventing web client graphing involving clients with fully qualified domain names
- Enable VDSO/faster time reads on more Linux v3.13 kernels

v6.3.15

- Report GPS signal strength in dB Hz rather than percentage
- Improvements to “sourcecheck” algorithm to detect bad sources more quickly
- Implement CLOCK_REALTIME_COARSE on systems that support it
- Prevent incorrect alarms about device state changes with Spectracom Tsync PCIe devices in some configurations
- Update in-sync status on Solarflare hardware to support Onload-translated hardware timestamps in some configurations

v6.3.14

- Changes to handle latest RT kernel scheduler changes from RedHat
- Grandmaster alarms when inlet temperature is out of range (temperature lowered to 95F)
- Improved accuracy and clock modeling
- Changes to support Rubidium unit on TimeKeeper grandmasters
- Changes to assist in factory test/setup of Velasync
- Improved Solarflare card accuracy
- Handle some protocol violations that some buggy PTP implementations on cameras have
- Resolve rare rendering issue with timing graph legend in web GUI
- Support TACACS+ authentication on TimeKeeper Grandmasters (requires basepackage version 1.2)
- Ensure configuration is parsed if possible during service stop, so any options that guide shutdown logic are honored
- Fix for default route detection where needed in environments with tunneling enabled
- Resolve web GUI issue where raw offset data could be displayed when the user had disabled that metric
- On Grandmaster, ensure default gateway can just be set in the general network settings in addition to the per-interface settings
- Basepackage upgrade (version 1.2) for TimeKeeper Grandmasters available to address bash shellshock vulnerabilities (CVE-2014-6271, CVE-2014-7169, CVE-2014-7186, CVE-2014-7187)

v6.3.13

- Fix grandmaster bug that could cause it to not report a source state change
- Grandmaster will consider “holdover” state as a bad state and will fall back to other sources rather than staying in holdover even when TimeKeeper believes holdover is better
- Improved randomization to avoid delay request congestion when many clients are started at the exact same time
- When hardware RX timestamps work but not hardware TX timestamps on Solarflare cards use software for all to get better sync
- Faster sync for major/minor sources
- Raise threshold for reporting GPS signal interference on grandmaster to avoid false alarms
- Fix bug that could cause timezone to not update properly
- Web configuration of IPMI interface added
- Fix for problem introduced in previous version for Spectracom Tsync driver unload
- Remove per-device staggering for NIC oscillators - improves SolarCapture compatibility
- Better support for selecting clients by IP and netmask range in the web GUI
- Web GUI now displays and sorts clients by DNS name, if known
- timekeeper-config RPM spec file now solely provides /etc/timekeeper.conf, instead of managing shared ownership with the timekeeper RPM package
- Updates to resolve /proc filesystem issues with some 3.14-based kernels
- Prevent some configurations from suppressing SNMP alerts about errors in the timing quality of PTP clients
- If the TimeKeeper status information reported on the web interface is stale, note that it’s dated for the user
- print_licinfo now reports detail about beta releases at runtime if applicable
- Timing map in web interface now has controls to handle large maps better
- Added AVOID_IFACES parameter to avoid enabling features on troublesome NICs
- Allow broader set of password character ranges in the TimeKeeper GUI
- Display current network information (such as DHCP address data) for the user in the TimeKeeper GUI

v6.3.12

- Updates for latest Solarflare driver/firmware PPS support
- Improved support for Flareon Solarflare cards
- TimeKeeper can collect accuracy, software version and other information from non-TimeKeeper clients and display them
- Support for Symmetric active/passive mode to better support Windows 2003 clients
- Improvements to graphing tools in web interface - better zoom control with sliders, added option to refresh graph immediately
- Improvements to accuracy/stability of TimeKeeper Grandmaster
- Prevent kernel module from being deleted while trying to rebuild .ko to match host kernel, in case host does not have build tools
- Reduced CPU usage across the TimeKeeper application
- TimeKeeper Grandmaster now uses local system account details for authentication
- Improved graphing when time sources report nonsense values - for easier detection of nonsense
- Enable hardware timestamping on all packet types with Solarflare Flareon cards, even if no PTP sources or servers are configured

- Web client interface improved for faster performance, while allowing more clients to be plotted with more intuitive controls
- Added optional SERVENTP_IFACE parameter, allowing TimeKeeper to only respond to requests seen on the named interface
- Allow TimeKeeper config RPM to be built as a non-root user

v6.3.11

- Improvements in data report by *report_problem.sh* script
- Fix to unicast PTP implementation that caused clients to not interact with some grandmasters properly
- Increase PTP log interval value when serving time to handle some PTP clients that report missing announce messages too early (older version of Domain Time II)
- Improved management/reporting for legacy NTP clients
- Ability to view logs in web GUI farther in past along with performance improvements
- Enable VDSO/faster time reads on more Linux v3.10 kernels
- Update to allow using /dev/ttyUSB* as RS232 input device
- Grandmaster network card firmware changes (for new hardware)
- Grandmaster GUI support for adding, managing and removing VLAN interfaces

v6.3.10

- The monolithic timekeeper.spec file has been refactored into 3 separate RPM spec files
- Provide prebuilt RPMs for a subset of the supported distributions
- Provide more information in web status page - last update time, warn if there are no recent updates
- Work around Linux clock API issue on SLES 11 SP3 kernels
- During installation, the TimeKeeper service status is no longer disabled, so that during an upgrade any previous configuration is retained

v6.3.9

- Fixes for PTP BMC - in some circumstances UTC offset and other data would not update from PTP announce messages
- Improved tracking in environments where negative one way delay times are possible
- Fixes for some bugs in GCC that can manifest on SLES 11 SP3 kernels as delayed wakeups

v6.3.8

- Improved Spectracom PCIe device tracking/noise rejection in noisy PCIe environments
- report_problem.sh now collects satellite data from a local Spectracom TSync PCIe card as part of the system summary
- Fix memory leak on some new kernels when using PHC devices

v6.3.7

- Add support for optimized timing calls on 3.5.0/3.5.7 kernels
- Automatic PTP source recovery when upstream networking hardware loses multicast group membership state
- Avoid checking for Spectracom TSync driver in configurations that do not require it

v6.3.6

- Log failures when NTP server hostname resolution fails
- When ENABLE_RERESOLVE is set continue to run if initial DNS query fails
- Fix for automatic upgrade version comparisons
- Updates for latest generation grandmaster hardware system chassis status queries
- Support hardware timestamping on VLAN interfaces
- TimeKeeper Grandmaster appliances now support HTTPS

v6.3.5

- Initial support for RHEL 7 beta
- Fix to correctly start up and take over timing calls when some processors are offline

v6.3.4

- Improvements to “Time Map” for NTPd self-referencing sources
- Significant improvement in NTP accuracy with hardware timestampping
- Fix so that kernel module is properly recompiled when Linux kernel changes
- Improvements to web GUI - make better use of screen real estate and usability/clarity changes
- Support for RADIUS authentication on the grandmaster appliance
- Respond to NTP requests on same interface they came in on no matter what routing is - allows multiple interfaces on the same subnet
- Changes for grandmaster to not read time when reading GPS data - avoids occasional bad timestamps

v6.3.3

- Syscall performance improvement for systems with high processor counts and unsynchronized oscillators
- TimeKeeper now provides a walkable SNMP tree and MIB file, along with new separate trap definitions
- Data was added to PTP client quality trap to provide more context

v6.3.2

- Reduced CPU load on grandmaster with many thousands of clients

- Grandmaster watchdog will not trigger at low CPU load any longer
- Faster disciplining of NIC oscillators
- Do not show internal oscillators/time sources in map (PHC devices)
- Fix to prevent server disconnection warnings in web GUI when uploading large base package files over slow links
- PTP Unicast/Telecom profile changes to support some legacy client systems
- Detect and alert when non-grandmaster licenses are placed on TimeKeeper grandmasters
- Users can now enable or disable the SNMP service on TimeKeeper grandmasters from the web GUI

v6.3.1

- Correct rare lost hardware timestamps when changing IP addresses on grandmaster hardware
- Improvements to client and server unicast PTP to better support more hardware
- Handle low-rate PTP sources more accurately
- Improved cable delay configuration/accuracy for TimeKeeper grandmaster
- Improvements for better short-term stability and also holdover performance on grandmaster hardware
- Allow setting syslog/rsyslog destination hosts on grandmasters
- Grandmaster base package update to Linux kernel for newer grandmaster hardware
- Improved NIC oscillator disciplining
- Fix poor performance with very low NTP sync rates regression that worked in previous version
- Add ability to change how long logs are archived on the grandmaster from web GUI
- Fix for rare and incorrect reporting of “No GPS Signal” message on grandmasters
- Report more GPS reception data on grandmasters

v6.3.0

- Significant improvement to web configuration interface
- Improve source and client web graphing interface
- Update time map view to improve performance and make information more clear
- Do not run sourcecheck on sources that are in very tight sync to remove noise
- Detect and when possible recover from GPS jamming on TimeKeeper Grandmaster
- Support for TimeKeeper Pocket Grandmaster hardware
- Add support for GPIO devices for alternate PPS sources
- Improved performance when using Solarflare and other hardware timestamp NICs
- Improved accuracy for PPS output on Solarflare cards

TimeKeeper Version 6.2.x

v6.2.14

- Add ‘loguser’ to Grandmaster to permit accessing logs and monitoring state
- Use UUID filtering on Solarflare cards to reduce card load for better stability
- Performance improvements for Solarflare driver with RHEL 6.4, Centos 6.4 and Linux 3.X kernels
- Fixes to factory reset on grandmaster and improvements in ssh access
- Allow factory reset on grandmaster hardware from console
- Bug fix for possible hang when stopping TimeKeeper with a high-priority RT task running on RHEL 5.x systems
- Web GUI has more clean (human readable) status/state information about Spectracom TSync cards
- Add beta support for Fedora 19
- Updates to hardware and software end user license agreement
- More improvements for faster/more accurate disciplining of all NIC oscillators (more accurate PPS output on Solarflare cards)
- Remove error message for non-errors when reading second port of Solarflare cards on Centos 6.4
- Corrections to documentation

v6.2.13

- Faster/more accurate disciplining of NIC oscillators (more accurate PPS output on Solarflare cards)
- Improve accuracy when using low NTP sync rates (long intervals)
- Fix bug with major/minor time when major time source is very noisy
- More clear (human readable) status/state information on “Status” page
- Add support for Linux 3.9+ kernels
- If TimeKeeper module fails to load still start timekeeper web so appliance configuration works properly
- New user command *tkstatus* that prints status of TimeKeeper
- Fix bug that could delay unload if user opens */proc/timekeeper_control*
- Do not require /etc/issue to be present during installation
- Configuration option for server to enable/disable RFC 868 support added to web management interface

v6.2.12

- If WEB_MANAGEMENT_IP is specified, validate that the named IP is a valid local address. If it is, the web interface will only be accessible via that IP address. If not, TimeKeeper will emit a trap and the web interface will be available on all interfaces so the issue can be resolved.

- Add option to cause occasional re-resolving host names to switch between NTP servers via round-robin DNS
- Grandmaster self-test displayed on status page
- Improved logging behavior on grandmaster appliance - means faster log collection for support requests
- Collect more information when creating problem reports from the web interface
- Fix when attempting to automatically upload problem report and download it locally
- Send alert (SNMP, email, rsyslog) on some startup errors
- Intel 82576 specific fix: permit mixed-use NTP/PTP without disabling hardware timestamps
- Resolve raw and CSV file download issue with some versions of Firefox
- When running the PTP BMC algorithm, prefer sources that specify a lower priority1 and priority2 value in their announce messages

v6.2.11

- Option to allow missing followup messages to handle some network card bugs
- Grandmaster appliance improvements for handling greater load and more PPS inputs
- Fixes for consistent behavior across Linux versions and Linux schedulers resulting in better performance when under heavy load
- Improved self-test of grandmaster systems on startup
- Add support for TIME (RFC 868) protocol and millisecond extension
- Fix for cases where saving and applying network settings on Grandmaster would save settings but not restart networking
- Only warn about license expiration once per day
- Fixes for grandmaster chassis temperature/power supply state check

v6.2.10

- More reliable startup/shutdown when using isolcpus and heavily loaded systems

v6.2.9

- Processor affinity change for kernel threads - do not default to the last CPU on start. Instead, move directly to the configured processor right away.
- /etc/init.d/timekeeper init script locking to prevent concurrent execution.

v6.2.8

- Significantly less memory required for serving NTP now
- Improved beta support for 3.8 kernels
- Now all threads (not just performance critical threads) are bound to the configured CPU
- Improved Grandmaster/GPS/oscillator accuracy and holdover
- Prevent possibility of reporting incomplete chassis status data in Grandmaster web interface

v6.2.7

- Added additional timing call takeover safety checks - to ensure safe TimeKeeper startup and shutdown for systems under extremely high load
- Prevent possibility of CLOCK_MONOTONIC backward movement (by 1-2 microseconds) on newer Linux kernels. Other clocks were not at risk of backward movement
- Documentation updates
- On TimeKeeper Grandmaster appliances, validate that the web management interface remains on when updating the configuration

v6.2.6

- Fix management data collection time window problem introduced in v6.2.5 that can cause excessive memory and CPU consumption as well as network traffic. Upgrading from v6.2.5 is strongly recommended and any use of v6.2.5 is discouraged.
- Detect if a restart of the web tools indicates normal operation or an error, and warn the user if needed
- Fix potential segfault/crash in TimeKeeper management server

v6.2.5

- Workaround for hardware timestamp bugs in kernels used on Centos/RHEL 6.4
- Change packet size for PTP sync and delay request messages to interoperate with Juniper switches better
- Pin kernel threads on the CPU number specified in /etc/timekeeper.conf, or use the last CPU if not specified
- Interoperate with Arista switches better - wait longer (400ms) for followup messages
- Single, clear trap message if license issues are detected at startup regardless of cause - specifics are provided in log file

v6.2.4

- Grandmaster appliance requires the user to agree to the EULA on first login
- Add client-side SNMP clearing traps for when sync error thresholds are restored
- Update Symmetricom BC635/BC637 driver
- Improvement to frequency estimator and predictor for better handling of noisy time sources

v6.2.3

- Beta support for Linux 3.8 kernels, including hardware timestamps
- Solarflare hardware timestamping has been re-enabled by default since latest Solarflare firmware (v3.3.0-6269) fixes PTP stability problems
- Allow processor frequency up to 5.0GHz
- Support for transparent clocks (in partnership with Arista)
- Better sync quality from improved frequency modeling

- Handle cases where remote PPS-based devices are untrustworthy
- Added a read-only user capability so non-administrators can use the web console to view data
- Avoid potential issues with long SNMP trap messages
- PTP sources can be configured to collect management data
- Web interface data collection updates for Fedora 18
- Expose SYNTONIZEONLY option in web interface
- Collect more network device data with ethtool on systems with renamed or virtual network interfaces
- Better drag behavior in “Time Map” when user is zoomed in

v6.2.2

- Clearer error message in cases where management tool is not supported
- Better sync quality due to better noise rejection
- VERBOSE_NTP option for detailed NTP logging information (client and server)
- Documentation updates
- On older hardware and old distributions, accept invalid timestamp data from the network stack until TimeKeeper is active and can offer correct data to callers
- Do not report license type in /proc/timekeeper output, since authoritative information is available elsewhere
- Reduce disk usage of bug/error reporting script on grandmaster configurations

v6.2.1

- Improvements to PTP BMC when using a grandmaster that has multiple servers
- Performance improvements for PTP management message handling under heavy load
- Error message added when service is started/stopped as non-root user
- Performance improvement for clock_gettime(CLOCK_MONOTONIC,...)
- Fix length field in management GET requests
- Less overhead in management query/response handling
- Improved performance in socket handling
- Per-source CABLEDELAY configuration option available
- Reduce memory load in client browsers when viewing timing network map

v6.2.0

- Visual representation of timing network in “Time Map”
- Improved scaling of time network management and monitoring tools
- Improved security features and detection of bad/false sources
- Reduced memory footprint
- Improved efficiency - able to handle more clients/servers with lower CPU overhead while running
- NTP queries now originate from port 123 to interact better with some firewalls
- Increased maximum PTP sync rate
- Respect rate change requests from NTP servers - slow down query rate when requested
- Add system (linux) log message tab to web interface for grandmaster
- Support for Redhat 6.4 beta release
- Bug fix to implementation of PTP best-master-clock algorithm
- Add support for Solarflare PPS input
- Disable HW timestamp on Solarflare cards until updated firmware is available (this works-around firmware bug)

TimeKeeper Version 6.1.x

v6.1.13

- Performance improvement for clock_gettime(CLOCK_MONOTONIC,...)

v6.1.12

- When displaying grandmaster network interface information, provide the MAC address of each interface if known
- Prevent case where web client can be provided stale configuration data
- Avoid locking some web UI elements on Firefox

v6.1.11

- Workaround hardware bug with UTC update on some GPS PCIe cards
- Make rate of alerts from sync error threshold errors configurable

v6.1.10

- Allow hardware timestamp disabling to be controlled just by the configuration file and not dependent on card type

v6.1.9

- Fix memory leak
- Fix for disabling hardware timestamps on Solarflare and Broadcom cards

v6.1.8

- Enable VDSO/faster time reads on Linux v3.2 kernels (Ubuntu 12 for example)
- Disable VDSO properly on Linux v3.x kernels where appropriate - prevents incorrect timestamps when VDSO is not supported on those kernels

v6.1.7

- Track Spectratime unit more aggressively during frequency changes

v6.1.6

- Re-order /proc/timekeeper information so primary source number is always shown
- Option to upload new licenses and TimeKeeper release via web tool in Grandmaster configuration
- Always run web tools even if license is expired
- Fix for selecting a CPU for TimeKeeper to run on that was introduced in 6.1.5
- Fix for possible loss of management data from clients
- Fix PTP best-master-clock algorithm implementation in cases where two PTP boundary clocks used the same grandmaster. Previously this caused rapid switching of the client between the two boundary clocks
- Reduce bandwidth needs when using web management tools, particularly to assist users on slower and remote links
- Document SYNCERRORTHRESHOLD as it applies to PTP servers in the timekeeper.conf example file

v6.1.5

- Reduced network I/O when using web management tools, for better performance over slow connections
- Selectively enable management interface on a specific network address for access control
- Client monitoring offset range graph on web interface is now auto-updating over the last 4 hours of data
- Add LOWQUALITY flag to improve tracking of low quality sources such as NTPd
- Fix when NTP server response with KISS field that is unprintable/garbage - safe reporting
- Unicast delay responses from Grandmaster configuration were not unicast in some cases - fixed
- Fix bug in Linux kernel for high CPU load that can be caused by hrtimer/futex calls
- Management improvements for Grandmaster configuration
- Improvements in security check for bad time, bad frequency, GPS spoof attacks
- Grandmaster web monitoring/control improvements
- Windows 7 compatibility improvements
- Web interface does not even provide high level status information until login succeeds
- Warn when no time sources are declared
- Web console breaks configuration options down into sections - sources, servers, and options
- Provide support for web management tools on more distributions

v6.1.4

- Prevent potentially doubled text when viewing client data on web management console
- Force new process session when performing a restart
- Update web status page more consistently

v6.1.3

- Resolve transient web management page load problems, also speed up page load
- Support for hardware timestamps on Broadcom 1G and 10G cards
- Better performance over noisy WAN links
- Improved performance/stability of web console
- Web console client information is updated more frequently (all data from clients displayed), more useful initial page for system status
- Improved holdover accuracy when no quality sources are available
- Improved bad/malicious source detection
- Interoperability problems between PTPv1 and PTPv2 fixed
- Support for Ubuntu 12.04 and SuSE 12.1

v6.1.2

- More data for smoother client sync quality graphs in the management console
- Allow web management console to store user preferences regarding plotting
- Prevent a case where PTP followup messages might be seen but discarded
- Better time series labelling on source and client graphs
- Users can now set (and retain) preferred graph scalings to better compare data
- Installer update - when a license is already present and installed, retain/reuse the license correctly even across partitions

v6.1.1

- Improved performance of web GUI
- Improved accuracy for NTP
- Improved accuracy for pulse-per-second (PPS) sources (including RS232/serial sources)
- Autodetection of serial port I/O locations
- Detection of web browsers with incomplete websocket implementations, link to plugin to assist

- Version number change - do not pad version number with leading 0's to avoid confusion
- Trimble GPS source support
- Improved PTP Telecom Profile unicast client and server robustness
- Management GUI now provides a log tab with recent timekeeper log data
- VDSO supported on Fedora 14
- Support for emitting syslog events
- Improved hardware timestamp cross checking
- More consistent logging format across source and servers
- "No logging" option that stops logging timestamps

v6.1.0

- Faster time reads on RedHat Enterprise 6.x series MRG kernels (3.0.x kernels)
- Unconditionally create /var/log/timekeeperclients to avoid situations where logrotate did not work properly
- Added holdover mode that allows TimeKeeper clients to maintain low-drift levels when primary sources are lost that are normally only available to atomic clocks
- Option to disable/enable hardware timestamps
- PTP client configuration can auto-configure TTL in some cases if the user forgets to configure it properly
- TimeKeeper RPM .spec file would create automatic dependencies that were incorrect - fixed now
- Allow users to specify SNMP OID with SNMPTRAPOID
- Handle occasional errors on hardware timestamp NICs to continue operation
- On install, if /etc/timekeeper.conf exists, it is left in place, and a reference /etc/timekeeper.conf.stock is installed
- PTP boundary clocks can now forward on client sync quality information to any upstream TimeKeeper servers
- Feature additions to management GUI - break client sync summaries down by IP/mask groups, provide static graphs of offsets, display live offset graphs by IP/mask group, add live netdelay graphs, improvements to the other accuracy graphs
- Do not send SNMP/email sync quality traps on startup - only do so after the sync has crossed into the specified sync error threshold range, to prevent large numbers of traps on startup as the clock slews in
- Do not send SNMP/email sync quality traps at a rate of more than once every 5 seconds, to avoid trap floods
- Reduction of overhead when reading the time - time reads are faster now
- Further protection against applications trying to drive Linux clocks
- RPM spec file leaves config files in place if they're already installed and modified

TimeKeeper Version 6.0.x

v6.0.3

- Workaround for occasional lost PTP followup packets on Intel 82576 hardware
- Reference license/configuration files on Arista switches in persistent storage rather than in /etc/ and /opt/timekeeper
- More timestamp-related messages now fall under the VERBOSE_TIMESTAMPS config option
- Detect common failures on some grandmaster appliances and switch to other sources more quickly
- Faster detection/failover when sourcecheck detects a faulty time source
- report_problem.sh script now collects timekeeper_management log
- Check and warn about bad license states once every 2 hours instead of once every 24 to avoid license check issues when bad clocks confuse license state information on short term license files
- Ability to read hardware timestamps from Solarflare cards when Linux itself does not support them (older Linux kernels)
- Improvements for NTP when used on WAN with slow/erratic network links

v6.0.2

- Fixes for some Linux distributions in cases where no PTP interface is specified - always select default route now
- Fixes for rare startup condition while serving that could cause segv
- Add new configuration option VERBOSE_TIMESTAMPS for explicit timestamp logging activity
- Make SET_TIME_ON_STARTUP visible to the web management GUI
- Add new configuration option VERBOSE_TCPDUMP for easy logging of network activity

v6.0.1

- Fix for updated UTC offset for PTP clients - previously did not properly recognize leap seconds from some PTP grandmasters
- Support for configuring multiple PTP sources of the same grandmaster and domain
- Make management query rate configurable - new configuration variable MANAGEMENT_QUERY_INTERVAL
- Display ideal and current tickrate graphs of sources in the security tab of the management console
- Additional quality/health/security checks of time sources when sourcecheck is enabled
- PTP servers can trigger alerts if any of the clients report a sync quality that exceeds SYNCERRORTHRESHOLD
- When restarting, remove old TimeKeeper PTP client files after they have been logrotated

v6.0.0

- Problem/bug reporting tool can optionally upload data automatically
- Web based GUI for management/monitoring of clients and server
- Improved holdover stability when no time source is available

- Ability to reset some GPS cards that do not lock a signal reliably on startup
- TimeKeeper option to set the time immediately (unconditionally) on startup
- Enable VDSO/vsyscalls on more CentOS releases
- PTP management support - PTP server collects/displays performance and statistics of clients
- Full PTP unicast support (useful for multihop network connections)
- Logs are rotated more frequently by default to reduce disk footprint
- Correction of PTP UUID put on wire
- Internal changes to allow operation on networking hardware (switches and routers)
- Send traps if a specific source's sync is over a configurable threshold
- Improved accuracy and clock model for PCI bus cards
- TimeKeeper requires less overhead while running

TimeKeeper Version 5.1.x

v5.1.04

- Collect uptime/load average with report_problem.sh
- Updates for PTPv1 serving on alternate domains
- report_problem.sh uses a more specific filename based on time
- Suppress unneeded messages during install

v5.1.03

- Avoid possible but rare time takeover conflict at service start
- Support for email notification of TimeKeeper events
- Workaround for problems when other packages consume/delay PTP packets
- More verbose reporting when necessary tools are not available for install
- report_problem.sh adds /sbin to the path for data collection
- Install saves currently installed /etc/timekeeper.conf as /etc/timekeeper.conf.backup

v5.1.02

- Support for bc7xPCIe PTP card
- Disallow negative round-trip times to reject timestamps that are erroneous because of other applications consuming PTP packets (caused by openonload)

v5.1.01

- Customer specific features
- Watchdog TimeKeeper task in-kernel in case user-mode component crashes
- Allow setting PTP server TTL of multicast packets
- When starting up with no valid source TimeKeeper watchdog failover thread exited - now it will continue trying
- Documentation updates

v5.1.00

- Detection of clock-rate ramping to detect thermal events and correct them
- "Sourcecheck" feature will cross-check multiple sources and allow detection of bad, invalid or malicious time sources
- Fixes for installation with sudo (uid, euid updates)
- New license system - options, features and allowed runtime controlled by a license file rather than timekeeper binary
- Support for RedHat 6.2
- Clock will be set (or "stepped") if time is more than 5 seconds off. Previous threshold was 30 seconds.
- Greatly improved accuracy/stability. Especially with noisy links (WAN and wireless links)
- TimeKeeper CPU processing overhead reduced
- PTP delay request messages are sent for every sync received (additional accuracy improvement)
- NTP client allows sync rate adjustment
- Install provides a generated RPM spec file to ease deployments

TimeKeeper Version 5.0.x

v5.0.08

- Correct port number in PTP announce messages
- Only step (or explicitly set) the clock on startup to overcome erratic output from some clock sources when they are reset
- Enable VDSO/vsyscalls on Fedora Core 15
- Faster execution/lower overhead for clock_gettime on some platforms

v5.0.07

- Faster clock_gettime() on some distributions/kernels (RedHat 5.5 for example)
- Improved consistency in timetests measurements

- Enable VDSO/vsyscalls on Ubuntu 10.04
- IEEE 1588 protocol implementation fixes - response with TAI timestamp to TAI grandmaster, control field properly set, port enumeration begins at 1

v5.0.06
- Improved accuracy and precision of timetests program to test time read overhead
- Fix for occasional lost PTP delay request messages

v5.0.05
- Enable VDSO/vsyscalls on CentOS 5.6
- Enable VDSO/vsyscalls on SLES 10

v5.0.04
- Enable VDSO/vsyscalls on Fedora 13

v5.0.03
- No longer requires a static route for PTP multicast group
- Fix for multicast PTP grandmaster mode (introduced v5.0.02)

v5.0.02
- Allow setting unicast/multicast for PTP grandmaster (ptp delay response) and client (ptp delay request) messages
- Allow setting a CPU mask of CPUs to run TimeKeeper on
- Allow a larger difference between hardware and software timestamps before discarding hardware timestamps. Necessary for some cards that are slow to transmit.

v5.0.01
- Fix for logrotate putting empty data at the beginning of log files after rotating them

v5.0.00
- Configuration file format change
- Ability to track multiple time sources (up to 50) at a time
- PTP support for boundary clock mode and best master clock selection
- PTP Followup messages are sent much sooner (10's of microseconds) after sync messages now
- PTP boundary clock functionality added
- Reduced overhead of timekeeper use processes and kernel processes
- Better discarding of invalid and stray PTP packets when used in multi-domain PTP networks

TimeKeeper Version 4.19.x

v4.19.03
- Improve sync quality with rate change estimation algorithms
- Fix for vDSO gettimeofday() call. Previously it could report 1000000 uSec in the tv_usec field which was incorrect. This was due to an error in the normalization function used to correct the tv_sec and tv_usec fields. Only gettimeofday() and only when used with vDSO did this behavior show up.
- Cosmetic fix for startup values in logs for offset. Initial values could be incorrect in first timestamp as reported in log. They were not used for time correction, though.

v4.19.02
- Fixes for merging major NTP time with a minor source like a PPS, particularly when there are large offsets

v4.19.01
- Fixes for more than 16 CPUs
- Correct PTP grandmaster mode 48-bit to 64-bit OUI mapping
- Fixes for RedHat MRG install

v4.19.00
- Bug fix for some versions of RedHat Enterprise 5 that caused interval timers - sleep(1), futex() and similar, to pause for too long
- No longer move IRQs off the last CPU on the system but TimeKeeper does still move itself to the last CPU
- Fix for utility and test program 'baddate' so that it now works for both positive and negative offsets
- Allow PTP sync rate between 64/packets per second and 0.5/packets per second on client and server side. Fixes client behavior with higher sync rates and adds ability to configure rate to server.
- Improvements to the logic used to discard time updates (NTP, PTP or local source) that are believed to be too noisy.
- TimeKeeper data files will be rotated via logrotate. A default logrotate recipe is installed with the product.
- *plots/plot.sh* is provided for easy generation of timing information.
- Allow PTP domain to be specified on the server side via the PTPDOMAIN configuration option.
- PPS generation utility provided (ppsgen/ directory) to compare TimeKeeper-managed systems against a reference PPS.
- Screensaver no longer activates at TimeKeeper service start and stop.

TimeKeeper Version 4.18.x

v4.18.06

- Fix for install problems on some non-bus card hosts

v4.18.05

- Fixes for TSync bus card. Can now use preloaded/configured driver or internal driver.

TimeKeeper Basepackage

These release notes are specific to the TimeKeeper basepackage, which applies to TimeKeeper Enterprise Grandmasters.

v1.12 (requires TimeKeeper v7.2.14 or newer)

- Additional Linux kernel security updates for CVE-2017-5753 (Spectre), CVE-2017-5715 (also Spectre), CVE-2017-5754 (Meltdown)

v1.11 (requires TimeKeeper v7.2.8 or newer)

- Linux kernel security update for CVE-2017-5753 (Spectre), CVE-2017-5715 (also Spectre), CVE-2017-5754 (Meltdown)
- Linux kernel security update for CVE-2017-8890
- Note that VelaSync Grandmasters require TimeKeeper v7.2.14 and an updated VelaSync license

v1.10 (requires TimeKeeper v7.2.8 or newer)

- Update for new version of FSMLabs Enterprise Grandmaster

v1.9 (requires TimeKeeper v7.2.8 or newer)

- Apply kernel, glibc security update for CVE-2017-1000364, CVE-2017-1000366 (Stack Clash)

v1.8 (requires TimeKeeper v7.2.3 or newer)

- Apply kernel security update for CVE-2016-5195 (Dirty COW)
- Update 10/40G network driver and firmware

Support

For TimeKeeper support, please email support@fsmlabs.com with a summary of the problem.

When you first contact support@fsmlabs.com we will ask you to upload logs from your system to avoid back-and-forth to capture your environment, system state, the problem and so on. We encourage you to do that anytime you request support to speed resolution of your problems or questions. To make that is easy we provide a tool to do that.

You can collect logs and upload them automatically via FTP with either our GUI or command line.

GUI:

Login as admin, navigate to the 'Help' tab, click the checkbox to automatically upload via FTP and then click 'Collect TimeKeeper Logs'. That will take a few minutes at which time the system will upload the logs and offer you the chance to download the file yourself if that fails.

Command line:

Run this command on Linux/Solaris:

```
/opt/timekeeper/release64/report_problem.sh -u
```

Run this command on Windows:

```
C:\Program Files\timekeeper\release64\report_problem.bat -u
```

If you encounter an error that the directory is not writable, then you can use the -d option to specify a writable directory for report generation.

If you block FTP or that doesn't work for some reason then the script will place a file in /tmp and let you know about it (or C:\Program Files\timekeeper\tmp directory on Windows). You can upload that file to us via https://license.fsmlabs.com/upload

Please make sure to also email support@fsmlabs.com with details about the upload, including the file provided, and any background details about the issue at hand. The better FSMLabs understands your question, the faster support can respond.

1. This takes effect after restarting TimeKeeper
2. Changes here take effect immediately
3. The numbers refer to list position, not array index. Only matters when using -s to add or delete entries.
4. This takes effect after restarting the network (this also restarts TimeKeeper)