The Synchronization Experts.

MANUAL

PTP Track Hound

Version 2.1.2

February 2, 2024

Meinberg Funkuhren GmbH & Co. KG

Table of Contents

1 Imprint

Meinberg Funkuhren GmbH & Co. KG
Lange Wand 9, 31812 Bad Pyrmont, Germany

Phone:	+ 49 (0) 52 81 / 93 09 - 0
Fax:	+ 49 (0) 52 81 / 93 09 - 230
Website:	https://www.meinbergglobal.com
Email:	info@meinberg.de
Date:	December 4, 2023
Manual Version:	1.3

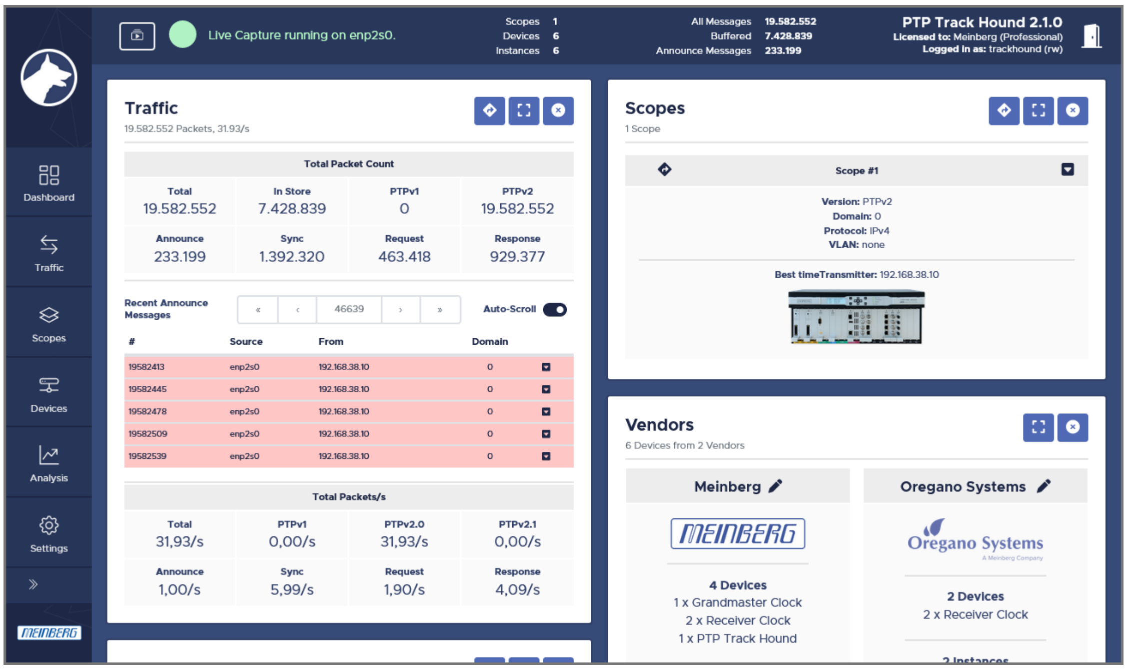
2 User Guide Revision History

Version	Date	Revision Notes
1.0	10/19/2022	Initial version
1.1	11/1/2022	– Updated to account for changes in PTP Track Hound v2.0.2 – Layout corrections – Standardized references to SSL/TLS certificates – REST API reference – Added information regarding the use of Npcap 1.70 under Windows – Added information regarding upgrading from v2.0.0 to v2.0.1 under Red Hat Enterprise Linux – Added information regarding executable permissions under macOS – Added additional details on logging levels – Added information about dashcam file links in Events window on Dashboard – Added information about segment metadata being bound to segment ID – Added additional screenshots in Chapter 7 and 8 for better context – Added icon illustrations – Added clarification about handling of PTPv2.1 clocks in existing PTPv2.0 scopes – Specified ability to upload and delete files manually from folder in Solo Mode and corrected Windows LOCALAPPDATA variable – More detailed explanation of Events window in Chapter 7 – Other general corrections and wording clarifications
1.2	12/21/2022	– Updated to account for changes in PTP Track Hound v2.0.4 – Added information on Raspberry Pi support – Added information on trial licenses – Minor corrections
1.3	2/1/2024	– Updated to account for changes in PTP Track Hound v2.0.6 and v2.1.0 – Updated screenshots (v2.1.0) – Updated information on licensing (from v2.1.1) – Monospace font now used to represent CLI and field inputs – Added information on updating an existing installation and on version update notification (from v2.0.4) – Reflected general changes in user interface behavior (from v2.1.0) – Added information on Capture Time Offsets (CTOs, from v2.0.5) in relation to initial setup wizard, logging, configuration, and reporting under "Scopes" – Added information on silent and unattended installations – Added illustration of "door" logout icon – Added new REST API endpoints as of v2.1.2 – Updated to account for other changes from PTP Track Hound v2.0.4 to v2.0.6 – Added alternative method of launching service under Linux – Changed standard port state terminology to IEEE 1588g-2022 recommendations (timeTransmitter, timeReceiver) – Changed all mentions of cookies in relation to locally stored data to "Web Storage" – Added cross-links in early chapters to other chapters for better understanding of as-yet unintroduced topics – Other minor corrections and layout adjustments

- Added chapter explaining the differences between measurement methods
- Added screenshots to Chapters 6.3 and 6.4
- New screenshot in Chapters 9 to better represent the Scopes page
- Rewritten Chapter 7 entirely; simplified greatly to account for the customizability of the Dashboard (from v2.1.0)
- Added information on new navigation features in Captured Packets tile (from v2.1.0)
- New chapter "Analysis" (from v2.1.0)
- Relocated information on graphs previously in "Scopes" section to "Analysis" section (from v2.1.0)
- Settings chapter (Chapter 11) updated to accommodate new configuration options for new features
- Renamed "panels" and "windows" to "tiles"
- Amended Chapter 12.9 ("Memory Usage") to reflect increase in allocable RAM and clarify minimum and maximum values for RAM and chunk size
- New chapter "Types of Analysis in PTP Track Hound v2" (Chapter 14.2)

3 Copyright and Liability Exclusion

4 Introduction to PTP Track Hound v2


Live Capture running on enp2s0.

Scopes 1
Devices 6
Instances 6

All Messages 19.582.552
Buffered 7.428.839
Announce Messages 233.199

PTP Track Hound 2.1.0
Licensed to: Meinberg (Professional)
Logged in as: trackhound (rw)

Traffic
19.582.552 Packets, 31.93/s

Total Packet Count			
Total	In Store	PTPv1	PTPv2
19.582.552	7.428.839	0	19.582.552
Announce	Sync	Request	Response
233.199	1.392.320	463.418	929.377

Recent Announce Messages « ‹ 46639 › » Auto-Scroll

#	Source	From	Domain
19582413	enp2s0	192.168.38.10	0
19582445	enp2s0	192.168.38.10	0
19582478	enp2s0	192.168.38.10	0
19582509	enp2s0	192.168.38.10	0
19582539	enp2s0	192.168.38.10	0

Total Packets/s			
Total	PTPv1	PTPv2.0	PTPv2.1
31,93/s	0,00/s	31,93/s	0,00/s
Announce	Sync	Request	Response
1,00/s	5,99/s	1,90/s	4,09/s

Scopes
1 Scope

Scope #1
Version: PTPv2
Domain: 0
Protocol: IPv4
VLAN: none
Best timeTransmitter: 192.168.38.10

Vendors
6 Devices from 2 Vendors

Meinberg
4 Devices
1 x Grandmaster Clock
2 x Receiver Clock
1 x PTP Track Hound

Oregano Systems
2 Devices
2 x Receiver Clock
2 Instances

PTP Track Hound v2 is a tool provided by Meinberg that can be used to easily log, review, and analyze PTP (IEEE1588) traffic in a given network. Unlike the widely used Wireshark, which serves more as a general-purpose packet sniffer for more conventional diagnosis and analysis, PTP Track Hound is specifically designed for the monitoring and analysis of PTP traffic, generating reports, charts, and illustrative diagrams for this purpose.

While generic traffic sniffers focus on communication between two devices, PTP Track Hound has been conceived from the ground up with an awareness that relationships between devices in a PTP network are only usefully understood in the context of an entire PTP network.

With Version 2, PTP Track Hound also introduces a new Web Interface that can not only be accessed from any browser, but allows the user interface to be accessed from any desktop PC via a standard web browser while the PTP traffic sniffer itself runs on a hub device through which all PTP traffic is known to pass through. This allows you to separate the capture point from the analysis workstation without needing to dump all UDP traffic to a network pipe.

Version 2 also introduces a new flexible licensing model with four licensing levels. The core functionality of PTP Track Hound remains free to use, and the Free version itself has been improved upon with various new powerful features. The Capture, Basic, and Professional licensing levels provide numerous additional enterprise-grade features designed for use in larger networks to meet professional monitoring and networking requirements. Further information on the licensing levels is provided by Chapter 5.2, "License Types".

This guide is provided to familiarize you with PTP Track Hound Version 2 and its features new and old, regardless of whether you are a networking professional who is new to the software, an experienced user of PTP Track Hound Version 1, or a hobbyist looking to learn more about PTP.

Users who have prior experience with PTP Track Hound Version 1 are advised to read Chapter 5.7, "Changes from Version 1".

And of course, if you need any assistance with using PTP Track Hound, we'll be happy to assist—just drop us a mail at **techsupport@meinberg.de** and we'll be in touch!

4.1 Terminology of Navigation Elements of the PTP Track Hound Version 2 Web Interface

The following terminology is used to describe the display and navigational elements that are employed in the PTP Track Hound Version 2 Web Interface:

The **Web Interface** (always capitalized) denotes the entirety of the PTP Track Hound configuration and monitoring interface accessible via a conventional web browser.

The **Header Bar** (always capitalized) is the navigation bar at the top of the page.

The **Sidebar** (always capitalized) is the bar located on the left of the page, containing links to the various sections.

Page refers to any complete page layout in the web browser, including Header Bar and Sidebar, as well as the contents of the section. It can also refer to any page that does not conform to the standard PTP Track Hound Web Interface layout (e.g., login page).

The **Content Area** (always capitalized) is the area in which all content is shown outside of the Header Bar and Sidebar.

Section refers to the five main sections listed in the Sidebar: **Dashboard**, **Traffic**, **Scopes**, **Devices**, and **Settings**.

Tile refers to a panel-like subsection within a page, denoted by a white background and rounded corners containing information or options. The display of tiles in any given section is typically configurable, such that they can be hidden or displayed, and in the case of the Dashboard section, also moved around.

Checkbox refers to any navigational element that can be enabled (denoted by a rounded square with a checkmark) or disabled (denoted by an empty rounded square).

Toggle switch refers to any navigational element that can be enabled or disabled. When disabled, it will be grayed out in order to avoid confusion as to which state is the "on" state.

Button refers to any element that is solely clicked on (using a mouse or touchpad) or pressed (on a touch display) to perform a given function.

Dialog box refers to any prompt that appears inside a page that renders the rest of the page inoperable until closed (for example, a file selection dialog box).

4.2 Formatting and Structural Principles of this Manual

This manual applies the following formatting and structural conventions:

Structure

This manual is designed to enable readers to set up PTP Track Hound v2 for use quickly and easily. All information to this end is provided by Chapters 5 (**Getting Started**) and 6 (**General Operation**)

The PTP Track Hound Web Interface is described section by section in greater detail in subsequent first-level chapters, specifically Chapters **7** (**Dashboard**), **8** (**Traffic**), **9** (**Scopes**), **10** (**Devices**), **11** (**Analysis**), and **12** (**Settings**).

Advanced information for application developers and introductory information for non-technical users is provided in the **Appendix**.

Formatting

Names of sections and tiles are displayed in **bold text**.

Field names, and button labels are also displayed in **bold text**. Example: **Add**.

Possible values and listed options for a configuration or status field are conventionally listed in *italics*.

URLs, field entries, and command line inputs are conventionally displayed in a `monospace typeface`. Example: The default username is `trackhound`.

References to other chapters in this manual are shown in dark blue and bold, and if the manual is viewed in a supported PDF reader, can be clicked on to directly jump to that chapter. Example: Chapter 4.2 "**Formatting and Structural Principles of this Manual**".

5 Getting Started

5.1 System Requirements

PTP Track Hound 2 can be run on a desktop or server PC running Windows, Linux, or macOS, or on a Raspberry Pi 3 or 4 running Raspberry Pi OS. The system requirements for each operating system are listed below.

Information:

These specifications assume the memory usage limit of 256 MB as configured by default for the storage of capture data. If this value is increased, the RAM requirements will increase accordingly.

Under special operating systems such as 'lightweight' Linux distributions running on low-performance systems, it is also possible to reduce the memory usage limit to reduce the RAM requirements.

Windows-Based PCs

Important!

Please note that PTP Track Hound v2 running under Windows only supports the detection or monitoring of PTP instances over an IPv4 or IPv6 (Layer 3) network. It does not support the detection or monitoring of PTP instances running directly over an IEEE802.3 Ethernet network (Layer 2).

- Microsoft Windows 10 Home 64-bit or better (32-bit Windows is **not** supported)
- Windows 10: 4 GB RAM, Windows 11: 8 GB RAM
- Any 1 GHz or better 64-bit CPU
- At least 300 MB of free disk space

Linux-Based PCs

Note: These specifications assume that you are using GNOME as a desktop environment. RAM requirements in particular will vary when using another or no desktop environment.

- Any 64-bit Linux distribution with at least Kernel 5.4 (e.g. Ubuntu 20.04 LTS, Linux Mint 20); 32-bit distributions are **not** supported
- 4 GB RAM
- Any 1 GHz or better 64-bit CPU
- At least 100 MB of free disk space

macOS-Based Systems

- macOS 11.3 Big Sur or better
- 8 GB RAM
- Any 1.6GHz or better 64-bit CPU
- At least 100 MB of free disk space

Raspberry Pi Systems

Important!

On Raspberry Pi 3B+ and Raspberry Pi 4 systems with just 1 GB RAM, adequate performance cannot be guaranteed under the desktop environment with the web browser open. We strongly recommend that systems with only 1 GB RAM be used as a dedicated, headless solution with a Basic or Professional license so that the Web Interface can be accessed from another device or the capture data can be forwarded to another PTP Track Hound v2 instance. Reducing the RAM allocation to 128 MB is also strongly recommended on systems with only 1 GB RAM.

- Raspberry Pi 3 Model B+ or better (Model A+ is **not** supported)
- Raspberry Pi OS Bullseye or better
- 1 GB RAM (ideally 2 GB RAM)
- At least 100 MB of free disk space

Tested Browsers

The PTP Track Hound v2 Web Interface has been successfully tested with the following web browsers:

- Mozilla Firefox, Version 105 or better
- Microsoft Edge, Version 105 or better
- Google Chrome, Version 105 or better
- Apple Safari, Version 14 or better
- Chromium, Version 105 or better

5.2 License Types

PTP Track Hound v2 is provided with four different licensing models to suit a variety of needs: **Free**, **Capture**, **Basic**, and **Professional**.

Important!

- When you purchase a Capture, Basic, or Professional License, you will be provided with three pieces of information: the licensee name, the license ID, and the license key. Please keep **all three** of these documented together in a safe place.
- Please note that license keys issued for the v2.0.x branch of PTP Track Hound v2 are only valid for features introduced in that branch and are separate from license keys issued for the v2.1.x branch.
- While v2.0.x license keys can still be used with PTP Track Hound v2.1.2, new features introduced in the v2.1.x branch will be disabled. Even so, an update to v2.1.2 is still recommended to benefit from bugfixes and security patches relating to existing features.
- Users with an active Service Agreement for PTP Track Hound v2 with Meinberg will have received their new keys for the v2.1.x branch automatically. An upgrade to v2.1.x from the v2.0.x can also be purchased separately.

Free License

The Free License is free of charge (subject to registration with Meinberg) and provides the fundamental capture functionality of PTP Track Hound v2. This is an incredibly powerful tool, so it would be an overwhelming task to list all of the Free version's functions here, but these features non-exhaustively include:

- Capture PTP traffic passing through one of the local device's network interfaces
- Export captured PTP traffic to a *pcap* file for archival or, if so desired, analysis in another tool
- Import of captured PTP traffic in a *pcap* file previously exported by PTP Track Hound v2 or another capture tool (such as WireShark)
- Dashcam Mode, where captured data from a defined number of seconds prior to an event is written automatically to a file when that event occurs
- Grouping of detected PTP clocks into "**scopes**" that provide an understanding of clock relationships
- Statistical analysis of PTP traffic to facilitate network load analysis
- Manual entry of metadata for each PTP device in the network

Capture License

The Capture License provides a cost-effective method of deploying a larger number of PTP Track Hound instances across an expansive network to collect and forward PTP traffic data to a central hub. It naturally provides all of the functions of the Free License, and also provides the ability to forward captured traffic to a PTP Track Hound v2 Professional instance.

Basic License

The Basic License naturally provides all of the functions of the Capture License, and also provides the ability to acquire data by sending PTP Management Messages, perform network measurements using NetSync Monitor, and record & compare the timestamps of PTP measurements using the Capture Time Offset functionality.

Professional License

The Professional License in turn provides all of the functions of the Capture and Basic Licenses, and also provides the following features:

- SNMP traps
- Email notifications for triggered events
- syslog support
- REST API for acquiring traffic and statistical data and also performing control and configuration processes on a PTP Track Hound Professional instance

Trial License

If you wish to try out the advanced features of the Basic or Professional versions of PTP Track Hound v2, it is also possible to request one or several time-limited trial licenses from Meinberg so that you can test it under real conditions before committing to a purchase. To request one or more trial licenses, send an email to **ptptrackhound@meinberg.de** with details of your requirements.

5.3 Installation

5.3.1 Installation under Windows

Information:

Please note that administrative permissions are required to install PTP Track Hound v2 under Windows.

1. If you have not already done so, download the executable installer `ptptrackhound_2.1.2_setup.exe` from the download page, for which you will have received a link by email following registration.

2. Launch the executable file. Click on "**Next**", review the User Agreement as appropriate, confirm your acceptance of the User Agreement, then click on "**Next**" again.

3. You will be prompted to either confirm the default installation path, or you may select a new installation path, either by entering it manually in the path field, or selecting it by clicking on the "**Browse...**" button. Once you are done, click on the "**Next**" button.

4. In the next step you will be prompted to specify where the shortcuts for the PTP Track Hound v2 apps should be placed in the Windows Start Menu. You can leave this at the default or define your own. Once you are done, click on the "**Next**" button.

5. You will then be prompted to specify whether you wish to create a desktop icon to launch PTP Track Hound v2's **Solo Mode**, (see Chapter 5.5.1), and whether to install the PTP Track Hound v2 capture service as a native Windows service. If you choose to do this, you may specify if the service should be started automatically when Windows is started, whether a self-signed SSL/TLS certificate should be generated for access to the Web Interface and REST API via HTTPS, and whether a default service configuration should be generated (using the Setup Wizard) during installation.

   The generation of an SSL/TLS certificate is only beneficial for PTP Track Hound v2 Basic, Capture or Professional instances; there is little benefit to HTTPS access with a Free License as the Web Interface can only be accessed from the device on which it is installed.

   An SSL/TLS certificate can also be generated later on using the command-line based `trackhound-certgen` tool; refer to Chapter 14.1, "**Operation from Command Line Interface**" for more information.

   The generation of a default service configuration during the installation process is recommended as the Setup Wizard will guide you step by step through the process of creating an initial functional configuration.

   If you decide not to install the native Windows service (by selecting "**Install binaries only**"), you will need to launch PTP Track Hound v2 each time after boot either from the command line manually or via a batch script.

6. If you opted to generate a self-signed SSL/TLS certificate for *HTTPS* access, you will now be prompted to enter the details for that certificate. If the folder already contains a previously generated certificate, you will be asked if you wish to use it or generate a new one.

If an SSL/TLS certificate is already present in the installation directory (e.g., when installing an updated version of PTP Track Hound v2 over an old version), you will be prompted to specify if you wish to keep this SSL/TLS certificate or generate a new one. If you choose to generate a new one, you will be prompted to enter the certificate's Common Name, Organization Name, and Validity.

The **Common Name** is the resolved name under which the PTP Track Hound v2 HTTPS service will be accessible. In a local network, this will usually be the local hostname, but may also include a local domain name to form a fully-qualified domain name if domains are used within the network.

Important!

If you intend to submit a Certificate Signing Request for your certificate, you should not use only an IP address as the Common Name, as most Certificate Authorities will reject this!

The **Organization Name** is generally used for the legal name of your organization. Unless you intend to submit a Certificate Signing Request for your SSL/TLS certificate, this entry is not required.

The **Validity** is the length of time in days that the certificate will remain valid for. When a certificate expires, the browser will issue a prominent warning that the certificate has expired and will require an exception to be registered for the Web Interface if you wish to continue visiting it. As a rule, you should strive to keep this period as short as reasonably possible, but shorter periods will entail more regular maintenance.

Important!

Applying shorter validity periods constitutes sound security practice but makes it necessary to generate new SSL/TLS certificates regularly using the `trackhound-certgen` tool, ideally before the old one expires. An expired SSL/TLS certificate will result in HTTPS access being disabled; if HTTP is also disabled in this case, this will result in loss of access to the Web Interface until the certificate is renewed or HTTP is re-enabled manually via the configuration file.

Refer to Chapter 14.1, "**Operation from Command Line Interface**" for more information.

7. If you are installing an updated version of PTP Track Hound v2 over an old one and already have a configuration file in the installation directory, you will be prompted to specify if you wish to keep this configuration file or generate a new one.

A summary of the options that you have selected will now be displayed. If you are satisfied that this is all correct, click on "**Install**" to begin the installation process.

8. During the installation process, you will be separately prompted to install **Npcap** and the **WebView2 Runtime** package.

Npcap is the packet sniffing tool that PTP Track Hound v2 relies on to capture PTP traffic passing through the device's network interface.

The **WebView2 Runtime** package is required to display the user interface for PTP Track Hound v2's Solo Mode.

Information:

If you are updating your PTP Track Hound v2 installation, it is recommended that you terminate any running PTP Track Hound service instances via **Services** under Windows beforehand.

If you do not, however, the PTP Track Hound v2 installer will attempt to terminate the service for you before performing the installation, but this process may overlap with the file installation. Should you receive an error message that `trackhound-service.exe` cannot be overwritten because it is in use, simply wait a moment and click on "**Retry**".

Important!

- PTP Track Hound v2 will not function without **Npcap** and it must be installed! Please do not skip the installation of this component!
- Please leave the options for driver administrator access and raw 802.11 traffic *disabled* as they are by default.

9. If you have purchased a Basic, Capture, or Professional License, you may enter the license information that you have been provided with here. If you have a time-limited trial license, please enter the expiration date provided with your license in the corresponding field in the format `YYYY-MM-DD`.

   If you wish to use the Free Version of PTP Track Hound v2, please leave all of these fields blank and click on "**Next >**".

Important!

The licensee name must be entered **exactly** as it is specified in the license information.

If you do not have this information to hand right away, you may skip this step and enter the license information later. However, please note that a Basic, Capture, or Professional License is required to access the PTP Track Hound v2 Web Interface from another device within your network; you will therefore need to enter the license information via the Web Interface from the device on which PTP Track Hound v2 is installed.

10. If you opted to set up a default configuration (or chose to overwrite an existing configuration), a console window running "`trackhound-config`" will now be opened to guide you through the initial setup of PTP Track Hound v2.

    Refer to Chapter 5.4, "**Initial Configuration Using Setup Wizard**" for more information.

11. The installation process is now complete. If you wish, select the corresponding checkbox to launch Solo Mode directly from here, or have the PTP Track Hound v2 service launched to enable access to the Web Interface via a web browser of your choice.

5.3.1.1 Unattended & Silent Installation

It is also possible to perform an unattended installation (for example, for automated rollouts to multiple devices) by executing the following from the command line (administrative permissions may be required):

```
\path\to\ptptrackhound_2.1.2_setup.exe /SILENT
```

If you wish to have an unattended installation performed completely silently (without any visible installation windows), launch the installer with the following:

```
\path\to\ptptrackhound_2.1.2_setup.exe /VERYSILENT
```

Any message boxes that may be displayed during the update process and would suspend the installation process until confirmed can be suppressed by passing the parameter */SUPPRESSMSGBOXES*.

Unattended or silent installations will install by default to the path:

```
%PROGRAMFILES%\Meinberg\PTP Track Hound 2
```

To specify an alternative installation path, pass the parameter:

`/PATH="\EXAMPLE\INSTALLATION\/PATH"`.

Information:

Please note that no configuration file is generated during a silent or unattended installation and the PTP Track Hound v2 service is not configured to launch automatically upon boot. Suitable configuration files will need to be generated manually using the `trackhound-config` tool (refer to Chapter 14.1, "**Operation from Command Line Interface**").

5.3.2 Installation under Ubuntu and Other Debian-Based Linux Distributions

PTP Track Hound v2 is provided both as a Debian package for use under Debian derivatives (most notably Ubuntu Linux).

These instructions assume that you are using Ubuntu Linux 22.04 LTS or Ubuntu Linux 20.04 LTS. They are also broadly applicable to non-LTS versions of Ubuntu, Linux Mint 21, Linux Mint 20, Debian Bookworm, Debian Bullseye, and Debian Buster, although these distributions are not officially supported.

Information:

These instructions assume that you are running PTP Track Hound v2 on a multi-user system without default root permissions. If you are already working in a root context, the use of `sudo` in these instructions is not necessary.

1. If you have not already done so, download the appropriate package from the download page, for which you will have received a link by email.

   Users of Ubuntu Linux 22.04 LTS, Linux Mint 21, or Debian Bullseye should download the *ubuntu-stable* package. Users of Ubuntu Linux 20.04 LTS, Linux Mint 20, or Debian Buster should download the *ubuntu-oldstable* package.

   Most of the dependencies for PTP Track Hound v2 will have been preinstalled under a standard Ubuntu Linux 22.04 LTS installation.

   For a comprehensive list of the dependencies of PTP Track Hound v2, enter:

   ```
   dpkg --info /path/to/package/ptptrackhound_2.1.2_amd64.deb
   ```

   All of these packages are available in the Ubuntu Linux repositories. You may install any missing packages manually beforehand using `apt-get` or Synaptic, or you can have them downloaded from the repository and installed automatically using the `--fix-broken` switch with `apt`.

2. Install the Debian package with the following command:

   ```
   sudo apt install /path/to/package/ptptrackhound_2.1.2_amd64.deb
   ```

   If you encounter unresolved dependencies, ensure that you have a working connection to your distribution's repository and use the command:

   ```
   sudo apt install --fix-broken /path/to/package
   /ptptrackhound_2.1.2_amd64.deb
   ```

3. If you intend to use the PTP Track Hound v2 Web Interface through your browser, the generation of a self-signed SSL/TLS certificate and key is strongly recommended to enable access via HTTPS. To this end, the tool `trackhound-certgen` is provided. For a full list of arguments accepted by the certificate generator, enter:

```
trackhound-certgen --help
```

For example, to generate a suitable certificate, you might enter:

```
trackhound-certgen -name ptptrackhound2 -o "Organization Name" -v 30
-f ~/th2-cert.pem -k ~/th2-key.pem
```

This will place a 30-day SSL/TLS certificate and key protecting the SAN `ptptrackhound2` in your `HOME` directory.

The **Common Name** (`ptptrackhound2`) is the resolved name under which the PTP Track Hound v2 HTTPS service will be accessible. In a local network, this will usually be the local hostname, but may also include a local domain name to form a fully-qualified domain name if domains are used within the network.

Important!

If you intend to submit a Certificate Signing Request for your certificate, you should not use only an IP address as the Common Name, as most Certificate Authorities will reject this!

The **Organization Name** is generally used for the legal name of your organization. Unless you intend to submit a Certificate Signing Request for your SSL/TLS certificate, this entry is not required.

The **Validity** (`30` in this case) is the length of time in days that the certificate will remain valid for. When a certificate expires, the browser will issue a prominent warning that the certificate has expired and will require an exception to be registered for the Web Interface if you wish to continue visiting it. As a rule, you should strive to keep this period as short as reasonably possible, but shorter periods will entail more regular maintenance.

Refer to Chapter 14.1, "Operation from Command Line Interface" for more information.

4. A configuration file must be generated before the PTP Track Hound v2 services can be launched. To do this, use the tool `trackhound-config`. For example, to generate a configuration file `/etc/ptpTrackHound2/th2-config.json`, enter:

```
sudo trackhound-config -o /etc/ptpTrackHound2/th2-config.json
```

5. The setup wizard will now guide you through the process of generating the initial configuration. Consult Chapter 5.4, "Initial Configuration Using Setup Wizard" and Chapter 14.1, "Operation from Command Line Interface" for more information.

6. You can now launch the PTP Track Hound v2 service in the current terminal (assuming the path specified in the example above) using:

   ```
   sudo trackhound-service -c /etc/ptpTrackHound2/th2-config.json
   ```

   Alternatively, you can run the service in the background with:

   ```
   sudo systemctl start ptpTrackHound2
   ```

   If you wish to have the PTP Track Hound v2 service launched automatically upon boot, enter:

   ```
   sudo systemctl enable ptpTrackHound2
   ```

5.3.3 Installation under Red Hat Enterprise Linux and Other RPM-Based Distributions

PTP Track Hound v2 is provided as an *RPM* package intended for use with Red Hat Enterprise Linux.

These instructions assume that you are using Red Hat Enterprise Linux 9 or Red Hat Enterprise Linux 8. They are also broadly applicable to Fedora Linux, although these distributions are not officially supported.

Information:

These instructions assume that you are running PTP Track Hound v2 on a multi-user system without default root permissions. If you are already working in a root context, the use of `sudo` in these instructions is not necessary.

1. If you have not already done so, download the appropriate package from the download page, for which you will have received a link by email.

   Users of Red Hat Enterprise Linux 9 should download the *rhel-stable* package.
   Users of Red Hat Enterprise Linux 8 should download the *rhel-oldstable* package.

   All of these packages are available in the Red Hat Enterprise Linux repositories. The installation tool `dnf` will resolve any unresolved dependencies automatically, but you may install these manually beforehand using `yum`.

   For a comprehensive list of the dependencies of PTP Track Hound v2, enter:

   ```
   dnf repoquery --requires /path/to/package/ptptrackhound-
   2.1.2-1.x86_64.rpm
   ```

2. Install the RPM package with the following command:

   ```
   sudo dnf install /path/to/package/ptptrackhound-2.1.2-1.x86_64.rpm
   ```

Important!

If you are updating an existing PTP Track Hound v2.0.0 installation to v2.1.2, please note that there is a bug in the v2.0.0 RPM package that will prevent v2.1.2 from installing properly. This can be fixed by entering the following after the RPM package is installed as above:

```
sudo dnf reinstall ./ptptrackhound-2.1.2-1.x86_64.rpm
```

This bug is fixed as of the v2.0.1 RPM package and will therefore not affect upgrades from v2.0.1 or later.

3. If you intend to use the PTP Track Hound v2 Web Interface through your browser, the generation of a self-signed SSL/TLS certificate and key is strongly recommended to enable access via HTTPS. To this end, the tool `trackhound-certgen` is provided. For a full list of arguments accepted by the certificate generator, enter:

```
trackhound-certgen --help
```

For example, to generate a suitable certificate, you might enter:

```
trackhound-certgen -name ptptrackhound2 -o "Organization Name" -v 30
-f ~/th2-cert.pem -k ~/th2-key.pem
```

This will place a 30-day SSL/TLS certificate and key protecting the SAN `ptptrackhound2` in your `HOME` directory.

The **Common Name** (`ptptrackhound2`) is the resolved name under which the PTP Track Hound v2 HTTPS service will be accessible. In a local network, this will usually be the local hostname, but may also include a local domain name to form a fully-qualified domain name if domains are used within the network.

Important!

If you intend to submit a Certificate Signing Request for your certificate, you should not use only an IP address as the Common Name, as most Certificate Authorities will reject this!

The **Organization Name** is generally used for the legal name of your organization. Unless you intend to submit a Certificate Signing Request for your SSL/TLS certificate, this entry is not required.

The **Validity** (`30` in this case) is the length of time in days that the certificate will remain valid for. When a certificate expires, the browser will issue a prominent warning that the certificate has expired and will require an exception to be registered for the Web Interface if you wish to continue visiting it. As a rule, you should strive to keep this period as short as reasonably possible, but shorter periods will entail more regular maintenance.

Refer to Chapter 14.1, "**Operation from Command Line Interface**" for more information.

4. A configuration file must be generated before the PTP Track Hound v2 services can be launched. To do this, use the tool `trackhound-config`. The service expects to find a configuration file by default at `/etc/ptpTrackHound2/th2-config.json`; therefore, to generate a corresponding configuration file, enter:

```
sudo trackhound-config -o /etc/ptpTrackHound2/th2-config.json
```

5. The setup wizard will now guide you through the process of generating the initial configuration. Consult Chapter 5.4, "**Initial Configuration Using Setup Wizard**" and Chapter 14.1, "**Operation from Command Line Interface**" for more information.

6. You can now launch the PTP Track Hound v2 service in the current terminal (assuming the path specified in the example above) using:

```
sudo trackhound-service -c /etc/ptpTrackHound2/th2-config.json
```

Alternatively, you can run the service in the background with:

```
sudo systemctl start ptpTrackHound2
```

If you wish to have the PTP Track Hound v2 service launched automatically upon boot, enter:

```
sudo systemctl enable ptpTrackHound2
```

5.3.4 Installation on Raspberry Pi

PTP Track Hound v2 is provided as a Debian package for use on Raspberry Pi systems (Model 3B+ or better).

These instructions assume that you are using Raspberry Pi OS (formerly "Raspbian"), Version 10 (Buster) or Version 11 (Bullseye). They are also broadly applicable to other Debian-based distributions that support the Raspberry Pi, such as Ubuntu Linux, although these distributions are not officially supported.

Information:

These instructions assume that you are running PTP Track Hound v2 on a multi-user system without default root permissions. If you are already working in a root context, the use of `sudo` in these instructions is not necessary.

1. If you have not already done so, download the appropriate package (32-bit or 64-bit) from the download page, for which you will have received a link by email.

   All dependencies for PTP Track Hound v2 will have been preinstalled under a standard Raspberry Pi OS installation.

   For a comprehensive list of the dependencies of PTP Track Hound v2, enter:

   ```
   dpkg --info /path/to/package/ptptrackhound_2.1.2_armhf.deb
   ```

   or

   ```
   dpkg --info /path/to/package/ptptrackhound_2.1.2_arm64.deb
   ```

   depending on which version you have downloaded.

2. Install the Debian package with the following command:

   ```
   sudo apt install /path/to/package/ptptrackhound_2.1.2_armhf.deb
   ```

   or

   ```
   sudo apt install /path/to/package/ptptrackhound_2.1.2_arm64.deb
   ```

   If you encounter unresolved dependencies for any reason, ensure that you have a working connection to the Raspberry Pi OS repository and use the command:

   ```
   sudo apt install --fix-broken /path/to/package
   /ptptrackhound_2.1.2_amd64.deb
   ```

3. If you intend to use the PTP Track Hound v2 Web Interface through your browser, the generation of a self-signed SSL/TLS certificate and key is strongly recommended to enable access via HTTPS. To this end, the tool `trackhound-certgen` is provided. For a full list of arguments accepted by the certificate generator, enter:

   ```
   trackhound-certgen --help
   ```

   For example, to generate a suitable certificate, you might enter:

   ```
   trackhound-certgen -name ptptrackhound2 -o "Organization Name" -v 30
   -f ~/th2-cert.pem -k ~/th2-key.pem
   ```

   This will place a 30-day SSL/TLS certificate and key protecting the SAN `ptptrackhound2` in your `HOME` directory.

   The **Common Name** (`ptptrackhound2`) is the resolved name under which the PTP Track Hound v2 HTTPS service will be accessible. In a local network, this will usually be the local hostname, but may also include a local domain name to form a fully-qualified domain name if domains are used within the network.

   Important!

   If you intend to submit a Certificate Signing Request for your certificate, you should not use only an IP address as the Common Name, as most Certificate Authorities will reject this!

   The **Organization Name** is generally used for the legal name of your organization. Unless you intend to submit a Certificate Signing Request for your SSL/TLS certificate, this entry is not required.

   The **Validity** (`30` in this case) is the length of time in days that the certificate will remain valid for. When a certificate expires, the browser will issue a prominent warning that the certificate has expired and will require an exception to be registered for the Web Interface if you wish to continue visiting it. As a rule, you should strive to keep this period as short as reasonably possible, but shorter periods will entail more regular maintenance.

   Refer to Chapter 14.1, "Operation from Command Line Interface" for more information.

4. A configuration file must be generated before the PTP Track Hound v2 services can be launched. To do this, use the tool `trackhound-config`. For example, to generate a configuration file `/etc/ptpTrackHound2/th2-config.json`, enter:

   ```
   sudo trackhound-config -o /etc/ptpTrackHound2/th2-config.json
   ```

5. The setup wizard will now guide you through the process of generating the initial configuration. Consult Chapter 5.4, "Initial Configuration Using Setup Wizard" and Chapter 14.1, "Operation from Command Line Interface" for more information.

6. You can now launch the PTP Track Hound v2 service in the current terminal (assuming the path specified in the example above) using:

```
sudo trackhound-service -c /etc/ptpTrackHound2/th2-config.json
```

Alternatively, you can run the service in the background with:

```
sudo systemctl start ptpTrackHound2
```

If you wish to have the PTP Track Hound v2 service launched automatically upon boot, enter:

```
sudo systemctl enable ptpTrackHound2
```

5.3.5 Installation under macOS

PTP Track Hound v2 is provided as an executable Shell script package for installation via the terminal.

1. If you have not already done so, download the appropriate package from the download page, for which you will have a received a link by email.

2. Install the package with the following command:

   ```
   sudo ./path/to/package/ptptrackhound_2.1.2_x86_64.run
   ```

   If you receive a "*command not found*" error message and you are certain that the path is correct, the executable bit of the installation package may not be set. In this case, please enter

   ```
   chmod +x ./path/to/package/ptptrackhound_2.1.2_x86_64.run
   ```

   and try to install the package again.

3. Once installation is complete, you will be prompted to specify whether the service should be started automatically when macOS is booted.

4. If you intend to use the PTP Track Hound v2 Web Interface through your browser, the generation of a self-signed SSL/TLS certificate and key is strongly recommended to enable access via HTTPS. To this end, the tool `trackhound-certgen` is provided. For a full list of arguments accepted by the certificate generator, enter:

   ```
   trackhound-certgen --help
   ```

   For example, to generate a suitable certificate, you might enter:

   ```
   trackhound-certgen -name ptptrackhound2 -o "Organization Name" -v 30
   -f ~/th2-cert.pem -k ~/th2-key.pem
   ```

   This will place a 30-day SSL/TLS certificate and key protecting the SAN `ptptrackhound2` in your `HOME` directory.

   The **Common Name** (`ptptrackhound2`) is the resolved name under which the PTP Track Hound v2 HTTPS service will be accessible. In a local network, this will usually be the local hostname, but may also include a local domain name to form a fully-qualified domain name if domains are used within the network.

   Important!

   If you intend to submit a Certificate Signing Request for your certificate, you should not use only an IP address as the Common Name, as most Certificate Authorities will reject this!

   The **Organization Name** is generally used for the legal name of your organization. Unless you intend to submit a Certificate Signing Request for your SSL/TLS certificate, this entry is not required.

The **Validity** (`30` in this case) is the length of time in days that the certificate will remain valid for. When a certificate expires, the browser will issue a prominent warning that the certificate has expired and will require an exception to be registered for the Web Interface if you wish to continue visiting it. As a rule, you should strive to keep this period as short as reasonably possible, but shorter periods will entail more regular maintenance.

Refer to Chapter 14.1, "**Operation from Command Line Interface**" for more information.

5. A configuration file must be generated before the PTP Track Hound v2 services can be launched. To do this, use the `trackhound-config` tool. For example, to generate a configuration file `/etc/ptpTrackHound2/th2-config.json`, enter:

```
trackhound-config -o /etc/ptpTrackHound2/th2-config.json
```

6. The setup wizard will now guide you through the process of generating the initial configuration. Consult Chapter 5.4, "**Initial Configuration Using Setup Wizard**" and Chapter 14.1, "**Operation from Command Line Interface**" for more information.

7. The PTP Track Hound v2 service can now be launched with:

```
launchctl start com.meinberg.ptpTrackHound2
```

5.4 Initial Configuration Using Setup Wizard

The PTP Track Hound v2 Setup Wizard guides you through the basic steps of configuring your PTP Track Hound v2 installation to ensure that it can be setup for use relatively efficiently.

All of these configuration parameters can be modified via the Web Interface or directly in the configuration file at any time after this initial configuration.

If you wish to run the Setup Wizard manually, execute the file "`trackhound-config`" from the directory in which PTP Track Hound v2 is installed.

Important!

If the PTP Track Hound v2 directory is in a protected directory and you wish to save the configuration file directly to that directory (e.g., "`C:\Program Files (x86)\Meinberg\PTP Track Hound 2`" under Windows, "`/etc/ptpTrackHound2`" under Linux) you will need to launch `trackhound-config` with administrative permissions.

1.
```
Start a live capture at service startup? (y/n)?
```

If answered with `y`, the PTP Track Hound v2 service will begin capturing PTP traffic immediately when the service is launched. If the service is installed as a native Windows service, this means that the data capture will begin as soon as the device is booted.

You will be prompted to specify for each network interface present in the system if traffic should be captured through that interface. If answered with `y`, PTP Track Hound v2 will prompt you for the following information and will listen for traffic on that interface:

- an optional alias by which you can recognize that interface
- an optional Segment ID to assist with organization of PTP traffic. Refer to Chapter 12, "Settings" for further information on segmentation,
- whether you wish to transmit PTP management messages to clocks detected on that interface to obtain more detailed data, and if so, the message interval in seconds (30–900 seconds, 60 seconds by default), whether to restrict management messages to a given PTP version (any, PTPv1, PTPv2, PTPv2.1), to a given PTPv2 domain or PTPv1 subdomain, and to a given network protocol (any, IPv4, IPv6, IEEE802.3), and the IPv6 multicast scope for management messages. Refer to Chapter 12, "Settings" for more information. This feature requires a **Basic License**.
- whether you wish to loop back management messages issued by PTP Track Hound v2 to the local loopback interface so that they too are captured by the local capture instance.
- whether you wish to enable hardware timestamping for more accurate capture timestamps (this prompt will only appear if your network interface supports hardware timestamping).

For more information on configuring network interfaces, refer to Chapter 12.1, "Packet Capture".

2.
```
Enable Capture Time Offsets (CTO)?
```

If answered with `y`, PTP Track Hound v2 will compare the timestamps of *Sync* and *Follow Up* messages received from a PTP clock against the timestamps provided by the PTP Track Hound v2 host device's own clock. The offset between these two timestamps is calculated and stored for statistical analysis. This feature requires a **Basic License**.

This feature is only beneficial if the clock of the host device on which PTP Track Hound v2 is running is itself synchronized, as otherwise there is no useful point of reference for the calculation of the offsets. Refer to Chapter 14.2, "Types of Analysis in PTP Track Hound v2" for further information.

3.
   ```
   Enable NetSync Monitor (NSM)?
   ```

   If answered with `y`, PTP Track Hound v2 will periodically transmit NetSync Monitor requests to identified or specified timeReceiver nodes in order to prompt them to transmit Sync and (in two-step mode) follow-up messages for the purpose of measuring time offsets. These measurements are stored for statistical analysis. This feature requires a **Basic License**.

   If enabled, you will be prompted to enter whether you wish to poll all known nodes detected by PTP Track Hound v2. In this case, any node that is detected by PTP Track Hound v2 will be polled.

   You will also be prompted to specify if you wish you to poll **specific** nodes. If you disabled polling of all known nodes in the previous step, this will allow you to effectively create a whitelist limiting which nodes should be polled with NetSync Monitor messages. If polling of all known nodes in the previous step was enabled, this prompt will allow you to 'force' polling of PTP nodes by their IEEE802.3, IPv4, or IPv6 address, even if PTP Track Hound v2 was otherwise unable to detect them.

   This feature is only beneficial if the timeReceivers polled with NetSync Monitor messages are synchronized. It also requires the nodes being polled to support NetSync Monitor. Refer to Chapter 14.2, "Types of Analysis in PTP Track Hound v2" for further information.

4.
   ```
   Receive traffic from remote trackhound-service instance(s) (y/n)?
   ```

   If answered with `y`, the local PTP Track Hound v2 instance will be able to receive PTP traffic captured remotely by another PTP Track Hound v2 instance.
   This feature requires a **Professional License**.

   You will be prompted to enter the IPv4 or IPv6 address and TCP destination port of the remote PTP Track Hound v2 instance, and also have the opportunity to enter an optional alias for that instance (for example, the name of the PC on which it is running), and to assign capture traffic sent by that instance to a specific Segment ID.

   Finally, you will be prompted to specify if the remote PTP Track Hound v2 instance is using AES-256 encryption to send its capture data and if so, the shared encryption key.

   You can configure as many external PTP Track Hound v2 instances to receive captured traffic from as you need. Once you are finished, answer "`n`" accordingly to the question "*Add another remote trackhound-service instance?*"

   For more information on configuring PTP Track Hound v2 to receive traffic from other Track Hound instances, refer to Chapter 12.1, "Packet Capture".

5.
   ```
   Allow remote trackhound-service instances to receive captured
   traffic from this trackhound-service (y/n)?
   ```

   If answered with `y`, the local PTP Track Hound v2 instance will be able to send PTP traffic captured by this PTP Track Hound v2 instance to another remote PTP Track Hound v2 instance for external analysis. This feature requires a **Capture License**.

   You will be prompted to enter the TCP source port for the remote connection; this is the port on which the receiving PTP Track Hound v2 instance will be **listening** for capture data.

   You will then be prompted to specify if you wish to add a list of clients that are permitted to receive capture data from this PTP Track Hound v2 instance. If answered with *y*, you will be prompted to enter the IPv4 or IPv6 address of each client, whether each client will be expecting AES-256-encrypted capture data, and the shared encryption key if so.

Important!

Please note that without a list of allowed clients, a PTP Track Hound v2 Professional instance will not be able to connect to this PTP Track Hound v2 Basic instance.

You will then be asked if you wish to disable the analysis of incoming traffic. This can reduce CPU usage (and thus power consumption) on devices that are only expected to forward captured data and will not themselves be analyzing any PTP traffic.

For more information on configuring PTP Track Hound v2 to forward traffic to other Track Hound instances, refer to Chapter 12.2, "Remote Capture".

6.
```
Capture initialization phase duration (off = 0, 0 - 300, leave empty
for default [30])
```

When the PTP Track Hound v2 capture service is first started, it will wait for the period of time specified here before it actually generates events and scopes based on newly detected or changed devices or instances. This ensures that any 'known' clocks & instances are detected upon startup, and helps ensure that events relate solely to the detection of 'new' clocks & instances. The default setting of 30 seconds will allow the system to 'settle' before PTP traffic is captured.

7.
```
PTP Track Hound will collect statistics on the offset and path delay
of some or all of the PTPv2 instances via Management Messages, CTO
or NetSync Monitor.
How many statistics entries do you want to keep in RAM per instance?

Input (5 - 15000, leave empty for default [300]):
```

If you have enabled measurements by means of Management Messages, Capture Time Offsets, or NetSync Monitor polling, this question will allow you to specify how many statistics entries should be retained in RAM for each PTPv2 instance.

8.
```
Do you want to set up thresholds for collected statistics metrics?
```

If answered with `y`, you will be presented with a menu containing ten options, selectable by entering a value of `0` to `9`. Selecting an option will bring up a prompt in which the event/alarm level should be selected, the threshold value in nanoseconds at which that event/alarm should be triggered, and whether this threshold should be restricted to specific instances.

The **Reported Offset** and **Reported Delay** relates to the offset values reported by other nodes in response to **Management Messages** and accordingly requires Management Messages to be enabled. The delta thresholds for these values monitor changes between two consecutive reported values, such that an alarm or event is triggered if the difference between two consecutive reported values exceeds this threshold. This allows alerts or reports to be issued regarding sudden and possibly unexpected changes in network conditions.

The **Measured Offset** and **Measured Delay** relates to the offset values measured by PTP Track Hound using **NetSync Monitor** and accordingly requires NetSync Monitor support to be enabled. The delta thresholds for these values monitor changes between two consecutive measured values, such that an alarm or event is triggered if the difference between two consecutive measured values exceeds this threshold.

The **Capture Time Offset** relates to the offset values between the timestamps embedded in incoming PTP messages and the associated timestamps generated by by PTP Track Hound based on the local clock for those messages and consequently requires Capture Time Offset support to be enabled. The delta thresholds for this value monitors changes between two consecutive recorded values, such that an alarm or event is triggered if the difference between two consecutive recorded values exceeds this threshold.

9.
```
Set up notifications via E-Mail (SMTP), SNMP or syslog (y/n)?
```

If answered with `y`, notifications regarding the triggering of defined events can be sent to an email address using a configurable SMTP smarthost, to a network management system via SNMP, or to a syslog server. This feature requires a **Professional License**.

If you decide to add an SMTP smarthost, you will be prompted to enter the smarthost address, the SMTP port (`25` by default), the sender email address, and any number of recipient email addresses. You will also be asked if you wish to enable smarthost authentication. Some mail relays will not forward mail without authentication.

If you decide to add a receiver for SNMP traps, you will be prompted for the address, SNMP port (`161` by default, and the SNMP version. If you are using SNMPv1 or SNMPv2c, you will be prompted for the SNMP community name. If you are using SNMPv3, you will be prompted for the SNMP engine ID, and the SNMP security name & level.

If you decide to add a syslog server, you will be prompted for the syslog server address, the server port (`514` by default), and the transport protocol (`UDP` or `TCP`).

You will then be prompted to specify if you wish to receive notifications for when data capture is started and stopped, when a new scope is detected (refer to Chapter 9, "**Scopes**" for more information), when a new device is detected (refer to Chapter 10, "**Devices**" for more information), when a new port is detected, when a new PTP instance is detected, when a port state is changed (e.g., a PTP clock changes from timeTransmitter to timeReceiver), when the local quality changes, when the Grandmaster quality changes, when the Grandmaster is changed, when a custom alarm is triggered or cleared, when inconsistencies arise in message sequence IDs, and when configured metrics (Management Messages, Capture Time Offsets, NetSync Monitor) cross thresholds or are normalized again.

For more information on configuring notifications in PTP Track Hound v2, refer to Chapter 12.3, "**Notifiers and Alarms**".

10.
```
Set up custom alarms (i.e.  Clock Class exceedance) (y/n)?
```

If answered with `y`, this will enable you to define custom alarms for the previously defined notification channels. This question will not appear if no notification channels are configured. This feature requires a **Professional License**.

Custom alarms are defined using REST API resource routes.

Please refer to Chapter 13, "**REST API Reference**" for more information.

You will be prompted to enter the REST API resource route, the comparison operator, the comparison operand, and the severity of the event, which will determine the level of the notification in the previously defined alarm channels. You may also add a custom alias for the alarm.

For more information on configuring notifications in PTP Track Hound v2, refer to Chapter 12.3, "**Notifiers and Alarms**".

11.
```
Use default mappings for PTP port states (y/n)?
```

If answered with `y`, PTP port states will use the IEEE 1588g-2022 terminology **timeTransmitter**, **Pre-timeTransmitter**, and **timeReceiver**.

If you choose not to use the IEEE 1588g-2002 terminology, you will be asked if you wish to specify your own terminology for these PTP port states. If you do not, the original IEEE 1588 terminology (**Master**, **Pre-Master**, and **Slave**) will be used.

This is a purely aesthetic change and does not affect the actual function of PTP Track Hound v2.

For more information on configuring terminology in PTP Track Hound v2, refer to Chapter 12.7, "**Terminology**".

12.
```
Maximum memory (RAM) usage in MB (1 - 8192, leave empty for default
[256])
```

This specifies the maximum amount of RAM that PTP Track Hound v2 will use to store PTP capture data. When this buffer is nearly full, older data will be purged from memory to make space for the newer capture data.

13.
```
Chunk allocation size in MB (1 - 16, leave empty for default [16])
```

This specifies the chunk sizes for the capture data RAM allocation. Larger chunk sizes will reserve more RAM at a time as needed. Smaller chunk sizes will reduce RAM usage by only allocating that chunk size, but the reallocation of RAM may cause performance to suffer slightly.

For more information on configuring memory usage in PTP Track Hound v2, refer to Chapter 12.9, "**Memory Usage**".

14.
```
Enable REST API and web interface via HTTPS (y/n)?
```

Information:

Before answering this question with `y`, please ensure that you have already generated the requisite SSL/TLS certificate and key file. If you have not, the Web Interface can be used or the configuration file can be manually edited to configure HTTPS access after the Setup Wizard has been completed.

This specifies whether the Web Interface and REST API will be accessible via HTTPS. This feature requires a **Professional License**. If answered with `y`, a SSL/TLS certificate will need to be generated; the Windows installer provides the means to do this automatically, while Linux and macOS users will need to do this manually using the tool `trackhound-certgen` (refer to Chapter 5.3, "**Installation**" for more information).

If you choose to enable HTTPS, you will be prompted to specify the port. The default value is `443`; note that the port must be specified manually when using the REST API or accessing the Web Interface via HTTPS on a different port.

15.
```
Enable REST API and web interface via HTTP (y/n)?
```

This specifies whether the Web Interface and REST API will be accessible via unencrypted HTTP. This feature requires a **Professional License**.

Important!

If you have not properly configured HTTPS or are unsure if it is properly configured, it is strongly recommended that HTTP be left enabled, as disabling HTTP while HTTPS access is not functional will prevent access via the Web Interface. HTTP can be disabled at a later time if necessary via the Web Interface or by manually editing the configuration file.

If answered with `y`, you will be prompted to enter the port number for HTTP access. The default value is 8080; note that the port must be specified manually when using the REST API or accessing the Web Interface via HTTP on a different port.

16.
```
Enable file server for capture and image file uploads (y/n)?
```

This enables you to define a file path on your device or network for the storage of imported capture files and for image files used to depict servers and vendors. Under Windows, this can be a mapped local or network drive or a UNC network path.

Please note that the PTP Track Hound v2 Setup Wizard will not create new folders if any part of the path does not yet exist. Any folder for the file server must be created beforehand.

17.
```
Set up user account(s) for web UI and REST API (y/n)?
```

This allows you to create user accounts for the Web Interface and REST API. You may also specify if each given account has write access; an account without write access cannot modify the configuration, upload capture files or images to the file server, stop or start the capture service, export capture data, or register a new license.

Regardless of whether you choose to set up accounts here, the default account (username: `trackhound`, password: `wood`) will be set up.

18.
```
Set up logging?
```

If answered with `y`, you will be prompted to specify whether you wish to direct log output to the `stdout/stderr` stream of the `trackhound-service` process, a defined text file, and either syslog or the Windows Event Log. In each case you will be prompted to specify the maximum severity level to be recorded in each log channel.

Important!

Whenever the PTP Track Hound v2 service is restarted or logfile function is disabled and re-enabled, the existing logfile will be overwritten. If you wish to preserve log output from previous sessions, consider using a local or remote *syslog* solution.

19.
```
Enter license information?
```

This prompt will only appear under Windows if license information has not already been provided during the installation.

If you have purchased a Basic or Professional License or have obtained a time-limited trial license, answer this question with `y` and enter the license information that you have been provided with here.

If you wish to use the Free Version of PTP Track Hound v2 or wish to enter your license data at a later time, answer this question with `n`.

Important!

The licensee name must be entered **exactly** as it is specified in the license information.

If you do not have this information to hand right away, you may skip this step and enter the license information later. However, please note that a Basic or Professional License is required to access the PTP Track Hound v2 Web Interface from another device within your network. You will need to enter the license information via the Web Interface from the device on which PTP Track Hound v2 is installed.

20.
```
Save the new configuration to a file (y/n)?
```

This question will only appear if the Setup Wizard has been called without specifying a filename for the configuration to be saved under (`-o`) or if it has been called as part of the Windows installation process.

If answered with `y`, you will be prompted to enter the path and filename under which the configuration will be saved.

If answered with `n`, the setup wizard will output the configuration to the console to be manually copied to an existing or new PTP Track Hound v2 configuration file.

5.5 Launching and Logging In

The primary method of interaction with PTP Track Hound v2 is the Web Interface, which can be accessed either via the integrated web browser of "Solo Mode" or via your own chosen web browser.

Web Interface via Browser

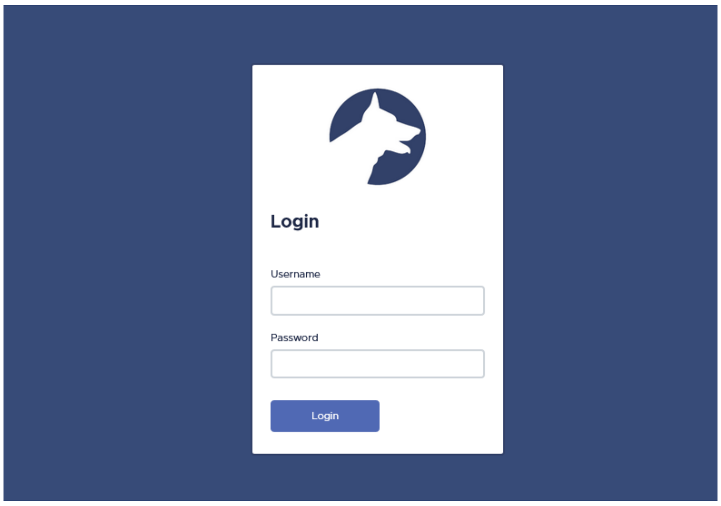


Figure 5.1: PTP Track Hound v2 Login Page

Ensure that the PTP Track Hound v2 service has been launched as described in Chapter 5.3, "**Installation**".

If you are accessing PTP Track Hound v2 from the device on which it is installed, open the URL `http://127.0.0.1:8080`. If you have disabled HTTP access and only permit HTTPS access, open the URL `https://127.0.0.1:443` instead.

If you have specified alternative ports for HTTP or HTTPS during the setup process, amend the URL accordingly.

If you wish to access PTP Track Hound v2 from another device that is reachable from the PTP Track Hound v2 installation, open the corresponding IP address as the URL with the corresponding port from your browser (e.g., `http://192.168.178.40:8080` or `https://192.168.178.40:443`).

Important!

A registered **Capture**, **Basic**, or **Professional** License is required to access the PTP Track Hound v2 Web Interface from a device other than the one that it is installed on.

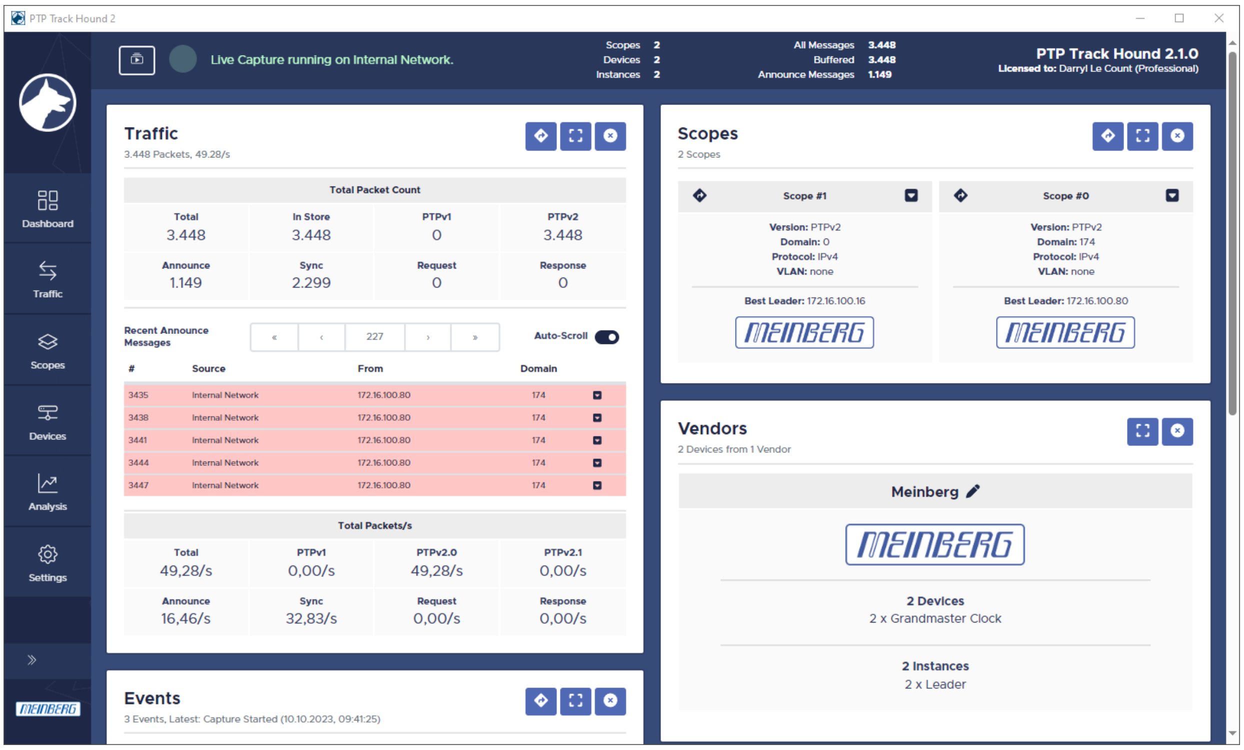
With a **Free** license, you can only access the Web Interface via the local loopback address `127.0.0.1` from the device on which it is installed.

Accessing the Web Interface via HTTPS/TLS is recommended for more secure access; however, the SSL/TLS certificate generated during the installation process is self-signed and you will therefore be prompted by your browser to add an exception for the PTP Track Hound v2 Web Interface.

Once the browser page has loaded, you will be presented with the PTP Track Hound v2 login page. Use the default credentials to log in at this point:

Username: `trackhound`
Password: `woof`

5.5.1 Solo Mode



Live Capture running on Internal Network.

Scopes 2
Devices 2
Instances 2

All Messages 3.448
Buffered 3.448
Announce Messages 1.149

PTP Track Hound 2.1.0
Licensed to: Darryl Le Count (Professional)

Traffic
3.448 Packets, 49.28/s

Total Packet Count			
Total 3.448	In Store 3.448	PTPv1 0	PTPv2 3.448
Announce 1.149	Sync 2.299	Request 0	Response 0

Recent Announce Messages « ‹ 227 › » Auto-Scroll

#	Source	From	Domain
3435	Internal Network	172.16.100.80	174
3438	Internal Network	172.16.100.80	174
3441	Internal Network	172.16.100.80	174
3444	Internal Network	172.16.100.80	174
3447	Internal Network	172.16.100.80	174

Total Packets/s			
Total 49,28/s	PTPv1 0,00/s	PTPv2.0 49,28/s	PTPv2.1 0,00/s
Announce 16,46/s	Sync 32,83/s	Request 0,00/s	Response 0,00/s

Events
3 Events, Latest: Capture Started (10.10.2023, 09:41:25)

Scopes
2 Scopes

Scope #1	Scope #0
Version: PTPv2	**Version:** PTPv2
Domain: 0	**Domain:** 174
Protocol: IPv4	**Protocol:** IPv4
VLAN: none	**VLAN:** none
Best Leader: 172.16.100.16	**Best Leader:** 172.16.100.80

Vendors
2 Devices from 1 Vendor

Meinberg

2 Devices
2 x Grandmaster Clock

2 Instances
2 x Leader

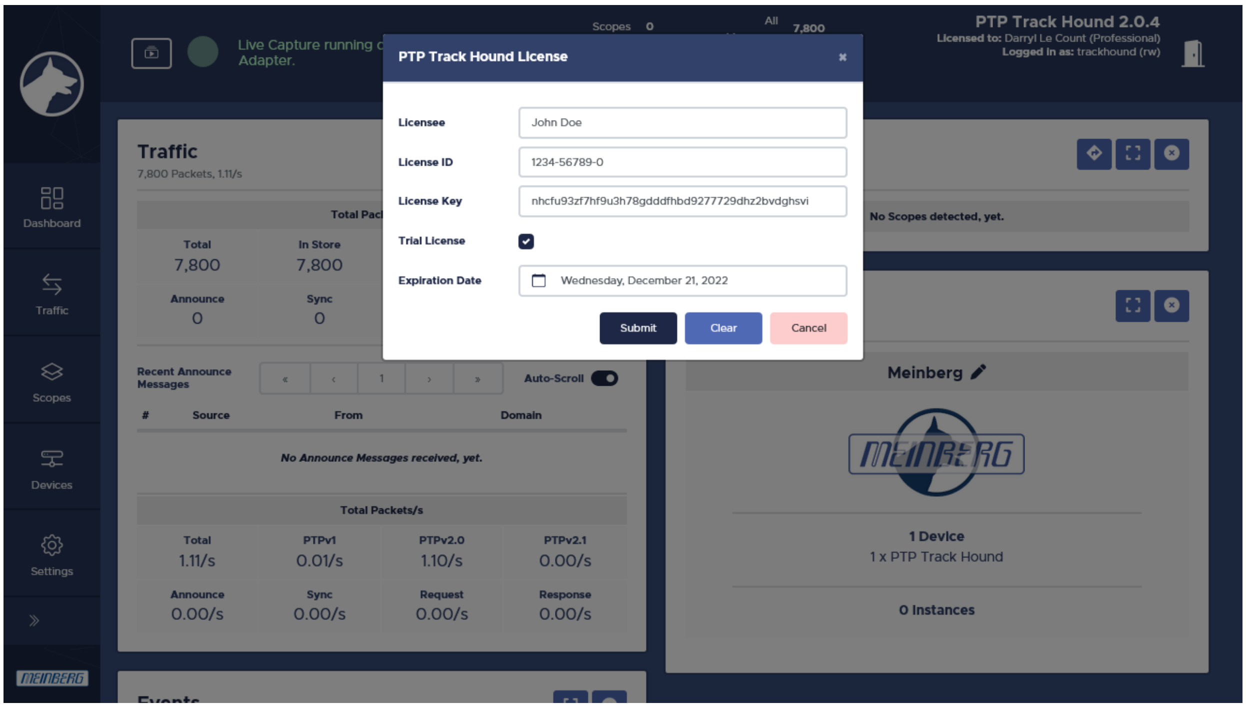
Figure 5.2: PTP Track Hound v2 Solo Mode

Solo Mode essentially replicates the experience that users of PTP Track Hound v1 may be familiar with. In Solo Mode, the user interface is displayed in its own window and the PTP Track Hound v2 service is launched alongside the user interface. Please note that the service will only be active while the user interface window is open; as soon as the window is closed, the service will be stopped and any ongoing traffic capture will cease.

To launch Solo Mode:

- Under Windows, select "**PTP Track Hound 2 Solo Mode**" from the Windows Start Menu.
- Under Linux, run `ptpTrackHound2` with root permissions (i.e., with `sudo` if necessary).
- Under macOS, launch *ptpTrackHound2* via Launchpad.

5.6 Activating or Swapping a License


PTP Track Hound License

Licensee	John Doe
License ID	1234-56789-0
License Key	nhcfu93zf7hf9u3h78gdddfhbd9277729dhz2bvdghsvi
Trial License	✓
Expiration Date	Wednesday, December 21, 2022

Figure 5.3: Registering a License

If, during installation, you did not activate your purchased PTP Track Hound v2 Capture, Basic, or Professional License (or your trial license) for any reason, or if you wish to amend one of your licenses (for example to move a license to another device, or to replace a trial license with a full license) or upgrade from a v2.0.x branch license to a v2.1.x branch license, you may (re-)activate or modify your license through the Web Interface.

Important!

Please ensure that you keep your v2.0.x branch license keys separate from your newer v2.1.x branch license keys. While v2.0.x license keys can still be used with PTP Track Hound v2.1.2, new features introduced in the v2.1.x branch will be disabled with an activated v2.0.x license. Follow the activation process below to upgrade your v2.0.x branch license to your v2.1.x branch license.

Following your purchase, you will have received three pieces of information by email, all of which you will need to activate your PTP Track Hound installation:

- The licensee name
- The license ID
- The license key

If you have received a trial license, you will also have received the expiration date of your license. Please make a note of this exact date as you will require this information to activate the trial license.

Activation of Your License

1. From the device on which PTP Track Hound v2 is installed, either open the Web Interface and log in, or alternatively launch Solo Mode. Both processes are described in Chapter 5.5, "**Launching and Logging In**".

2. At the top right of the page, beneath the PTP Track Hound title and version, you will see a link either entitled "**Unlicensed (Free Version)**" or beginning with "**Licensed to:**". Click on this link to open the **License Registration** dialog box.

3. You will be prompted to enter the licensee name, license ID, and license key, as provided following your acquisition of a PTP Track Hound v2 license.

   If your license is a trial license, you should also mark the checkbox labeled "**Trial License**" and use the calendar selector under "**Expiration Date**" to select the expiration date provided to you by Meinberg.

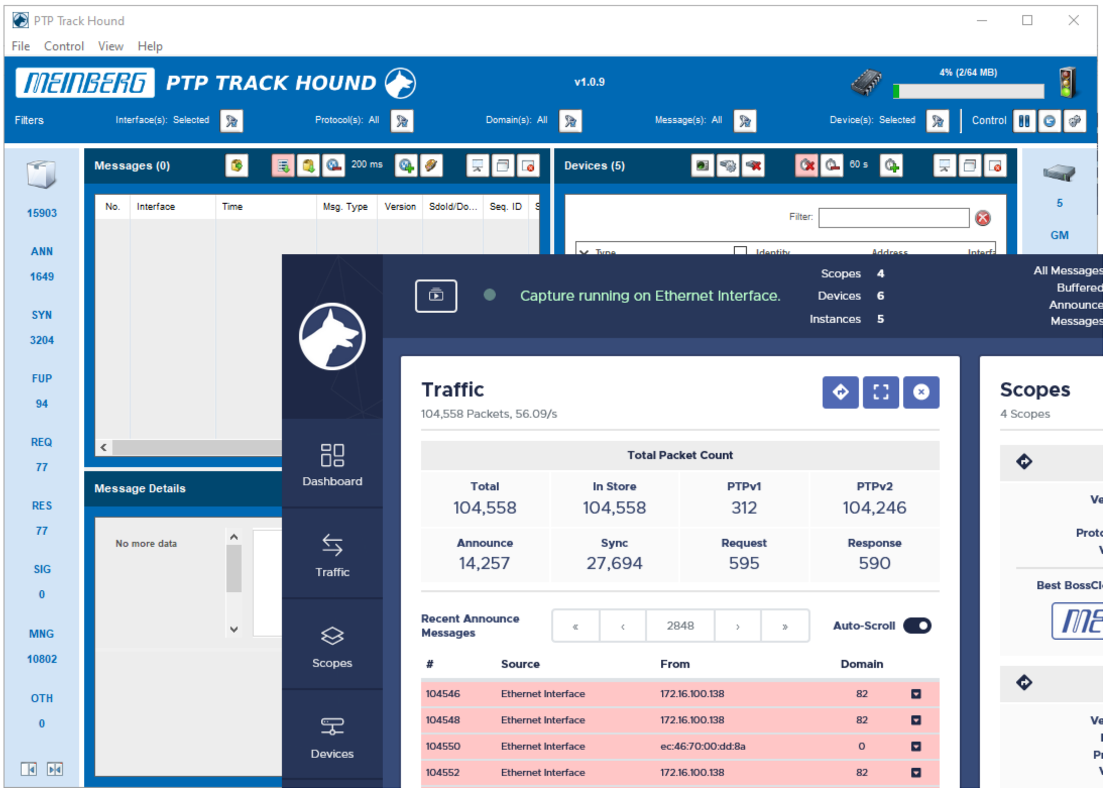
Important!

The licensee name must be entered **exactly** as it is specified in the license information. If you have problems registering your key and have copied the license information directly from the registration email, check that you have not inadvertently appended extra characters such as spaces to the beginning or end of the values.

This is especially important when registering a new license via the PTP Track Hound v2 Web Interface from another device than the one it is installed on, as incorrectly registering a new key will cause you to lose remote access to the Web Interface and you will need to access it again from the local device!

4. Click on "**Submit**". The license key dialog box will close and the license key link should change to the new licensee name and type.

5.7 Changes from Version 1



No.	Interface	Time	Msg. Type	Version	Sdold/Do...	Seq. ID

15903	
ANN	1649
SYN	3204
FUP	94
REQ	77
RES	77
SIG	0
MNG	10802
OTH	0
GM	5

Capture running on Ethernet Interface.

Scopes	4
Devices	6
Instances	5

Traffic
104,558 Packets, 56.09/s

Total Packet Count

Total	In Store	PTPv1	PTPv2
104,558	104,558	312	104,246
Announce	**Sync**	**Request**	**Response**
14,257	27,694	595	590

Recent Announce Messages « ‹ 2848 › » Auto-Scroll

#	Source	From	Domain
104546	Ethernet Interface	172.16.100.138	82
104548	Ethernet Interface	172.16.100.138	82
104550	Ethernet Interface	ec:46:70:00:dd:8a	0
104552	Ethernet Interface	172.16.100.138	82

Scopes
4 Scopes

Best BossCl

Some users may already be familiar with PTP Track Hound Version 1, and this chapter is intended for those individuals. Some of the user interface concepts will seem familiar to users of Version 1, even though Version 2 represents a considerable paradigm shift in the user experience.

The most obvious and significant change in the use of Version 2 is the support for a Web Interface accessible using any supported browser. For more information on the supported browsers, please refer to Chapter 5.1, "System Requirements". This method of operation allows the network capture service to be run independently of the Web Interface, so that it can continue to run in the background while the user interface is closed.

The Web Interface is intended to replace the monolithic approach of Version 1, where the capture service would be launched and terminated concurrently with the user interface. However, a "Solo Mode" is also provided with Version 2 for those who prefer the original monolithic application style; this mode opens the user interface via its own window and launches and terminates the capture service together with the interface window, broadly imitating the method of operation of Version 1.

The Dashboard of Version 2 is broadly comparable to the user interface of Version 1; the summarized PTP message statistics that were on the left of the Version 1 UI are now provided at the top left of the **Traffic** section.

The **Messages** table that listed every PTP message captured in Version 1 has been replaced on the central Dashboard by a simpler summary of the most recently received *Announce* messages; the familiar pastel-toned, multicolored list containing all captured PTP messages is now located under the **Traffic** section. The filtering of these messages in Version 2 works somewhat differently compared to Version 1 due to how domains and network protocols are grouped in this new version. For more information, refer to Chapter 8, "Traffic"

The detailed information provided by the **Message Details** panel of Version 1, specifically a byte-by-byte breakdown of the message, the Ethernet/IP/UDP header data, the PTP message header data, and the PTP message content, is now accessed either by expanding on one of the *Announce* messages in the summary provided on the Dashboard or one of the messages in the **Traffic** section. The information is organized in Version 2 more or less identically to Version 1.

The **Devices** panel as it existed under Version 1 no longer exists in Version 2. Whereas Version 1 would display the identified Grandmaster Clocks and its timeReceiver Clocks as subordinate to it, Version 2 treats PTP timeTransmitter/timeReceiver relationships in the context of **Scopes**, which fundamentally changes how clock relationships are represented. For more information on Scopes, please refer to Chapter 9, "**Scopes**".

The **Events** panel under Version 1, which provides notifications of potentially important changes to the PTP network (additions of new instances, changes in quality levels, etc.), is now found on the **Analysis** page. For more information on the Analysis page, please refer to Chapter 11, "**Analysis**".

Under the hood, one of the key additions to Version 2 is the ability to use one instance of PTP Track Hound to capture traffic and forward it to another instance running on another device in a network. In Version 1, traffic could only be captured via a network interface on the local device. Version 2 enables traffic to be captured close to the source in different subnets and then forwarded to a 'hub instance' for centralized processing.

Finally, because this new Version 2 of PTP Track Hound is focused on the provision of enterprise-grade remote monitoring features, it also now also features a multitude of logging and notification features, including user-definable alarm conditions and comprehensive SNMP trap, syslog, and email notification support.

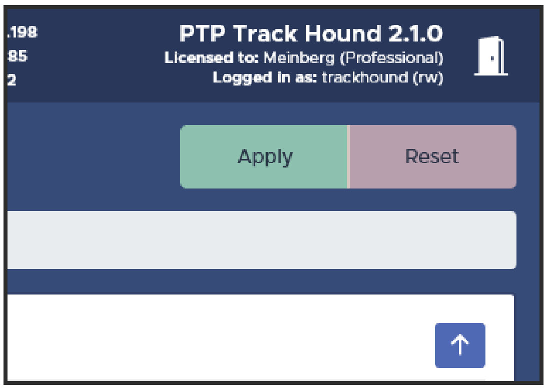
6 General Operation

6.1 Basic Configuration and Operation Principles

Depending on the type of data, some configuration data is stored directly on the PTP Track Hound v2 host (server-side storage), while other data is stored in the web browser's Local Storage (client-side storage).

Most of the persistent server-side changes to the configuration of PTP Track Hound v2 following the installation process are performed in the **Settings** section of the Web Interface. Whenever any changes are made to the server-side configuration, the green "**Apply**" button must be selected for them to be saved and applied.

To revert any changes to the configuration without applying them, either open another section of the Web Interface without clicking on "**Apply**", or click on the red "**Reset**" button.


.198
85
2
PTP Track Hound 2.1.0
Licensed to: Meinberg (Professional)
Logged in as: trackhound (rw)
Apply
Reset

Important!

Certain configuration tiles, namely the "**Terminology**" and "**User Management**" tiles, provide their own green "**OK**" and red "**Cancel**" buttons that control the entry of new terms and usernames. Please note that even when "**OK**" is clicked to confirm an entry, you will still need to click on "**Apply**" at the top right of the page to apply and save the configuration.

In addition to all settings made in the "**Settings**" section of the Web Interface, any metadata manually entered in relation to detected clocks and assigned segments is also stored on the PTP Track Hound host device.

Visualization preferences in the PTP Track Hound v2 Web Interface, specifically hidden tiles, are stored by your web browser locally in your browser's Local Storage. Therefore, please note that deleting your browser's web storage will also clear those settings and restore the default view configuration (in which all tiles are visible).

Each tile in a given section has various buttons, colored white with a blue background, that perform specific navigation functions:

The "**Up Arrow**" button scrolls the page back to the top.

The "**Turn Right Arrow**" button navigates to the corresponding section with more detailed information relating to what is shown in the tile.

The "**Full Screen**" button expands a given tile to encompass the entire page, hiding all other tiles in that section.

The "**Leave Full Screen**" button conversely restores the tiles as they were previously displayed before full screen mode was selected.

The "**Close**" button hides the tile.

The "**Collapse**" button minimizes content in that part of the tile, while the corresponding "**Expand**" button displays it again.

The "**Reset View**" button at the bottom right of the page (or top right of the page if all tiles are closed) restores the default tile arrangement by displaying any tiles that were previously closed. This button is only displayed if there are tiles hidden.

The "**Closed Windows**" button at the bottom right of the page (or top right of the page if all tiles are closed) allows you to select a hidden tile to be restored to the page. This button is only displayed if there are tiles hidden.

6.2 Live Capture of PTP Traffic in Network

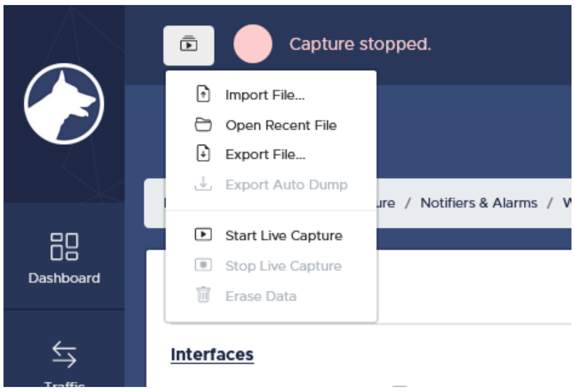
The core function of PTP Track Hound v2 lies in its ability to capture live PTP traffic from any or all of the network interfaces of the devices on which PTP Track Hound v2 is running and to dynamically analyze it.

Depending on how the PTP Track Hound v2 service was configured during the initial configuration process, the live capture may be initialized automatically when the service is started. When you open the Web Interface, the pulsing green circle next to the **Menu button** at the top left indicates that a live capture is running. A red, non-pulsing circle indicates that the current data capture relates to a previous live capture but that the live capture currently is not running, while a yellow, non-pulsing circle indicates that the capture data relates to an imported capture file.

If the live capture has not been started automatically (or has been stopped), it can be manually (re)started from the Web Interface by clicking on the **Menu button** at the top left and selecting "**Start Live Capture**".

If you wish to stop the live capture at any time for any reason (for example, to ensure that the capture data can be exported to a file cleanly), click on the **Menu button** and select "**Stop Live Capture**".

When the live capture is stopped and restarted again, new data will be appended to the existing capture data. If you wish to erase the capture data and start the live capture anew, click on the **Menu button** and select "**Erase Data**".



6.3 Importing pcap Capture Files

PTP Track Hound v2 is capable of importing externally or locally exported *pcap* capture files. These may be generated by the local PTP Track Hound v2 instance, another PTP Track Hound v2 instance, or a third-party traffic capture tool such as Wireshark.

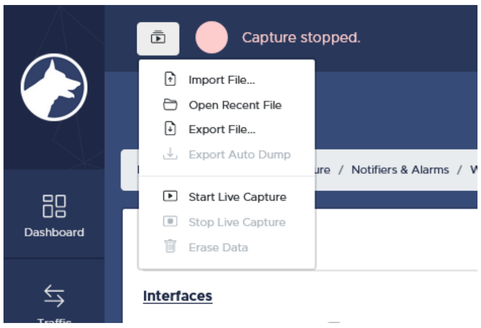
The File Server must be enabled and a valid file server path must be specified before you can import *pcap* capture files.

Ideally, capture data from Wireshark should be exported as a *pcap* file, not a *pcapng* file. However, *pcapng* files renamed to the file extension `.pcap` should be read without problems by PTP Track Hound v2.

If the *pcap* capture file contains non-PTP traffic, it will be filtered out by PTP Track Hound v2.

Important!

Any live capture that is running will be terminated when you import a *pcap* file.



1. Click on the **Menu button** at the top left of the page and select "**Import File**".

2. Select the *pcap* file that you wish to import via the file selector that appears.

Important!

Please note that if the newly imported *pcap* file has the same name as a file already on the PTP Track Hound v2 file server, the file on the file server will be overwritten without any warnings or prompts!

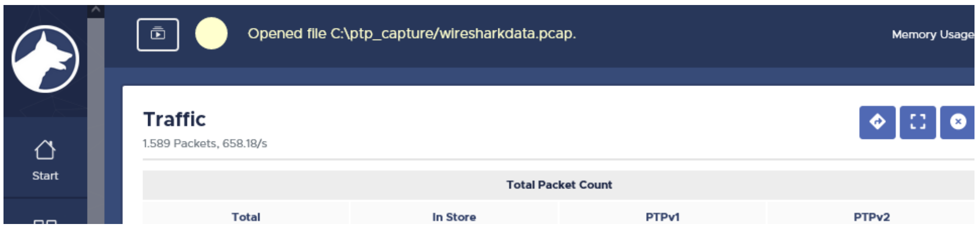


Figure 6.1: A successfully imported pcap file

3. If the file is successfully imported, you will see a prompt in the Header Bar that it has been opened (see Figure 6.1).

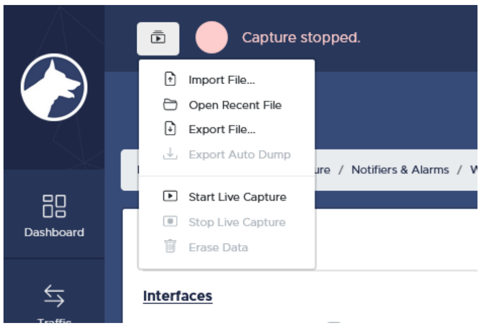
4. The captured data is now ready for review using the Traffic, Scopes, Devices, and Analysis sections.

6.4 Exporting pcap Capture Files

PTP Track Hound v2 is capable of exporting *pcap* capture files of its live captures and also re-exporting imported capture files for use on other PTP Track Hound v2 instances or with third-party tools such as Wireshark.

Information:

It is strongly recommended that any running live capture be stopped before a *pcap* file is exported. If an export is attempted while a live capture is running, PTP Track Hound v2 will continue to append data to the file while it is still being received, which may result in the capture file export taking a very long time (or, if large volumes of data are being generated, may result in the export never completing).



1. If you wish to re-export an existing *pcap* file, load it first by clicking on the **Menu button** at the top left of the page, selecting "**Open Recent File**" and selecting the corresponding file.

2. Click on the **Menu button** at the top left of the page and select "**Export File**".

3. Use the file selector to select the path and filename that you wish to save the capture file under.

6.5 Changing Password

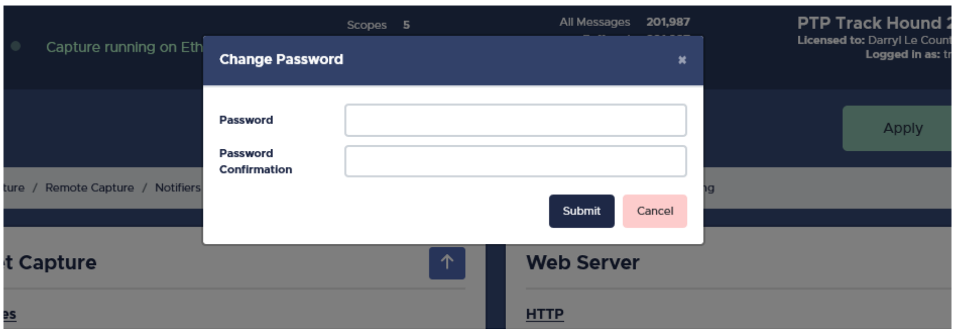


Figure 6.2: Changing the Password

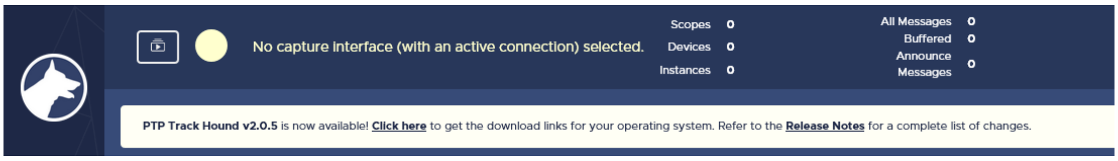
Users can change their own user password by clicking on the "**Logged in as...**" link at the top right of the page.

This opens a prompt in which the user can enter the new password in the first field and confirm it in the second. Click on "**Submit**" to apply the change or "**Cancel**" to close the prompt without making any changes.

Information:

It is also possible for any user with Write Access to change any other user's password. This is done in the **Settings** section. Refer to Chapter 12, "Settings" for further information.

6.6 Updating PTP Track Hound v2



PTP Track Hound v2.0.5 is now available! Click here to get the download links for your operating system. Refer to the Release Notes for a complete list of changes.

Figure 6.3: Notification that a New Version of PTP Track Hound v2 is Available

Whenever a new version of PTP Track Hound v2 is published on the PTP Track Hound v2 website, you will be notified as shown in Figure 6.3 above. Clicking on one of the links provided will allow you to review the Release Notes for the new version or download the full installer or package for the new version.

Please note that there are no specific update packages; both new installations and updates to existing installations are performed using the same package.

6.6.1 Updating an Installation under Windows

A PTP Track Hound v2 installation under Windows follows broadly the same procedure as a new installation as described in Chapter 5.3.1, "**Installation under Windows**"; simply execute the installer file downloaded from the PTP Track Hound download page in the same way.

However, there are certain important aspects to consider during updates:

- The installation directory must be identical to the original installation. Therefore, if you have specified an installation directory that is not the installer default, the installer must be manually pointed towards the directory of the original installation.
- The installer will allow you to create a new SSL certificate and corresponding key file, but will prompt you to confirm if you wish to overwrite any certificate and key file already present in the directory under the specified installation directory under the name of `th2-cert.pem` and `th2-key.pem` respectively.
- The installer will only detect your existing configuration file and prompt you to confirm if you wish to overwrite it if the existing file is named `config.json` and is located in the specified installation directory. If your configuration file has a different location or name, it will not be detected by the installer, but PTP Track Hound v2 can of course continue to use it if the configuration file is specified via the command-line parameters upon launch.

It is also possible to perform an unattended update by executing the following from the command line (administrative permissions may be required):

```
\path\to\ptptrackhound_2.1.2_setup.exe /SILENT
```

If you wish to have an unattended update performed completely silently (without any visible installation windows), launch the installer with the following:

```
\path\to\ptptrackhound_2.1.2_setup.exe /VERYSILENT
```

Any message boxes that may be displayed during an unattended or silent update process and would suspend the installation process until manually confirmed can be suppressed by additionally passing the parameter `/SUPPRESSMSGBOXES`.

Important!

Unattended or silent installations will install by default to the path:

`%PROGRAMFILES%\Meinberg\PTP Track Hound 2`

If your existing installation is at another location, this must be specified using the `/PATH` parameter:

`/PATH="`*`\EXAMPLE\INSTALLATION\PATH`*`"`.

6.6.2 Updating an Installation under Ubuntu and Other Debian-Based Distributions

Your PTP Track Hound v2 installation can be updated using the Debian package downloaded from the PTP Track Hound website in the same way that it was installed, using the following command:

```
sudo apt install /path/to/package/ptptrackhound_2.1.2_amd64.deb
```

This process will not affect your configuration or certificate files.

6.6.3 Updating an Installation under Red Hat Enterprise Linux and Other RPM-Based Distributions

Your PTP Track Hound v2 installation can be updated using the RPM package downloaded from the PTP Track Hound website in the same way that it was installed, using the following command:

```
sudo dnf install /path/to/package/ptptrackhound-2.1.2-1.x86_64.rpm
```

This process will not affect your configuration or certificate files.

Important!

If you are updating an existing PTP Track Hound v2.0.0 installation to v2.1.2, please note that there is a bug in the v2.0.0 RPM package that will prevent v2.1.2 from installing properly. This can be fixed by entering the following after the RPM package is installed as above:

```
sudo dnf reinstall ./ptptrackhound-2.1.2-1.x86_64.rpm
```

This bug is fixed as of the v2.0.1 RPM package and will therefore not affect upgrades from v2.0.1 or later.

6.6.4 Updating an Installation on Raspberry Pi

Your PTP Track Hound v2 installation can be updated using the Debian package downloaded from the PTP Track Hound website in the same way that it was installed, using the following command:

```
sudo apt install /path/to/package/ptptrackhound_2.1.2_armhf.deb
```

or

```
sudo apt install /path/to/package/ptptrackhound_2.1.2_arm64.deb
```

This process will not affect your configuration or certificate files.

6.6.5 Updating an Installation under macOS

Your PTP Track Hound v2 installation can be updated using the executable Shell script package downloaded from the PTP Track Hound website in the same way that it was installed, using the following command:

```
sudo ./path/to/package/ptptrackhound_2.1.2_x86_64.run
```

If you receive a "command not found" error message and you are certain that the path is correct, the executable bit of the installation package may not be set. In this case, please enter:

```
chmod +x ./path/to/package/ptptrackhound_2.1.2_x86_64.run
```

This process will not affect your configuration or certificate files.

6.7 Logging Out

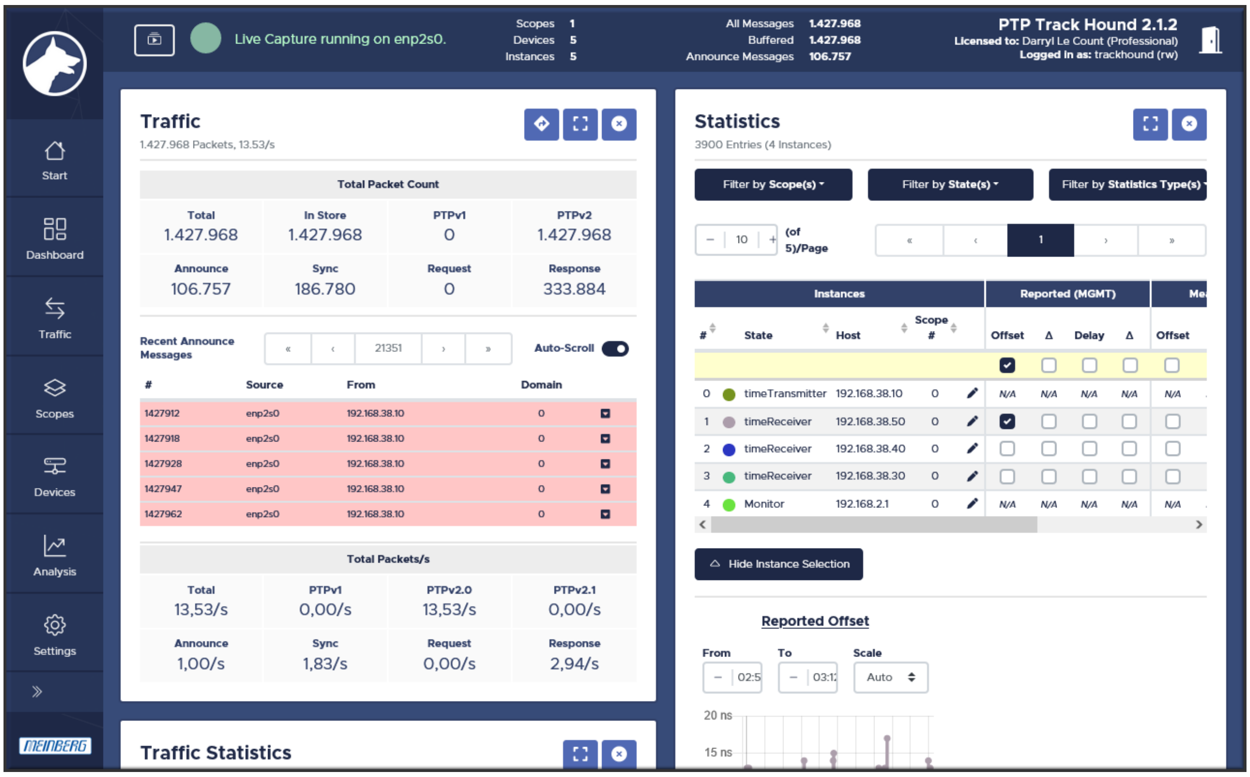
Logging Out via Web Browser

To log out of the PTP Track Hound v2 Web Interface, click on the door icon at the top right of the page. This will return you to the login screen. If the capture service was left running at the time of logging out, it will continue to run in the background.

Closing Solo Mode

To terminate PTP Track Hound 2 in Solo Mode, you simply need to close the window. This will automatically terminate the PTP Track Hound v2 capture service.

7 Dashboard



Live Capture running on enp2s0.

Scopes 1
Devices 5
Instances 5

All Messages 1.427.968
Buffered 1.427.968
Announce Messages 106.757

PTP Track Hound 2.1.2
Licensed to: Darryl Le Count (Professional)
Logged in as: trackhound (rw)

Traffic
1.427.968 Packets, 13.53/s

Total Packet Count			
Total 1.427.968	**In Store** 1.427.968	**PTPv1** 0	**PTPv2** 1.427.968
Announce 106.757	**Sync** 186.780	**Request** 0	**Response** 333.884

Recent Announce Messages « ‹ 21351 › » **Auto-Scroll**

#	Source	From	Domain
1427912	enp2s0	192.168.38.10	0
1427918	enp2s0	192.168.38.10	0
1427928	enp2s0	192.168.38.10	0
1427947	enp2s0	192.168.38.10	0
1427962	enp2s0	192.168.38.10	0

Total Packets/s			
Total 13,53/s	**PTPv1** 0,00/s	**PTPv2.0** 13,53/s	**PTPv2.1** 0,00/s
Announce 1,00/s	**Sync** 1,83/s	**Request** 0,00/s	**Response** 2,94/s

Traffic Statistics

Statistics
3900 Entries (4 Instances)

Filter by **Scope(s)** Filter by **State(s)** Filter by **Statistics Type(s)**

10 (of 5)/Page « ‹ 1 › »

Instances					Reported (MGMT)				Me
#		State	Host	Scope #	Offset	Δ	Delay	Δ	Offset
					✓	☐	☐	☐	☐
0	●	timeTransmitter	192.168.38.10	0	N/A	N/A	N/A	N/A	N/A
1	●	timeReceiver	192.168.38.50	0	✓	☐	☐	☐	☐
2	●	timeReceiver	192.168.38.40	0	☐	☐	☐	☐	☐
3	●	timeReceiver	192.168.38.30	0	☐	☐	☐	☐	☐
4	●	Monitor	192.168.2.1	0	N/A	N/A	N/A	N/A	N/A

Reported Offset

From 02:5 To 03:1 Scale Auto

Figure 7.1: Dashboard

The **Dashboard** provides a customizable overview of the captured traffic and the results of the analyses performed on this traffic. It allows tiles from any of the pages Traffic, Scopes, Devices, and Analysis to be added and moved around as desired to provide you with the information that you need at a glance.

Your personalized Dashboard starts out completely empty. To start adding tiles, click on the "**Add Tile**" button.

The top section of the window as shown in Figure 7.2 will contain the tiles that you add to the Dashboard. The bottom section contains a drop-down menu containing categories of tiles and the tiles available in the currently selected category.

Select a category from the drop-down menu, then click on the "**Add Tile**" button to add a tile. It will immediately appear in both the top area of the "**Customize Dashboard**" window and in the Dashboard itself behind it.

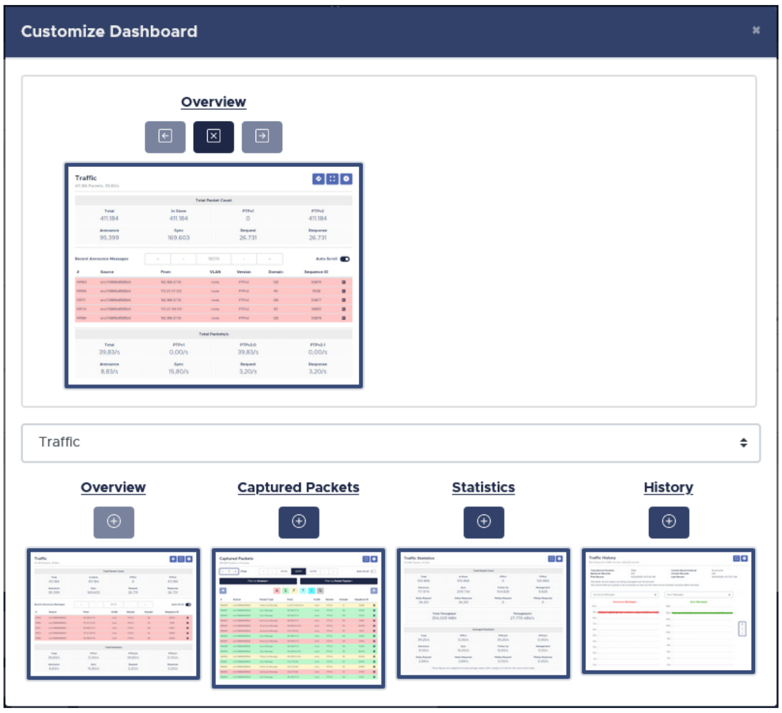


Figure 7.2: Customizing the Dashboard

Once tiles have been added to the Dashboard, the order of these can be moved around using the two arrow buttons or individual tiles can be removed using the "**X**" button.

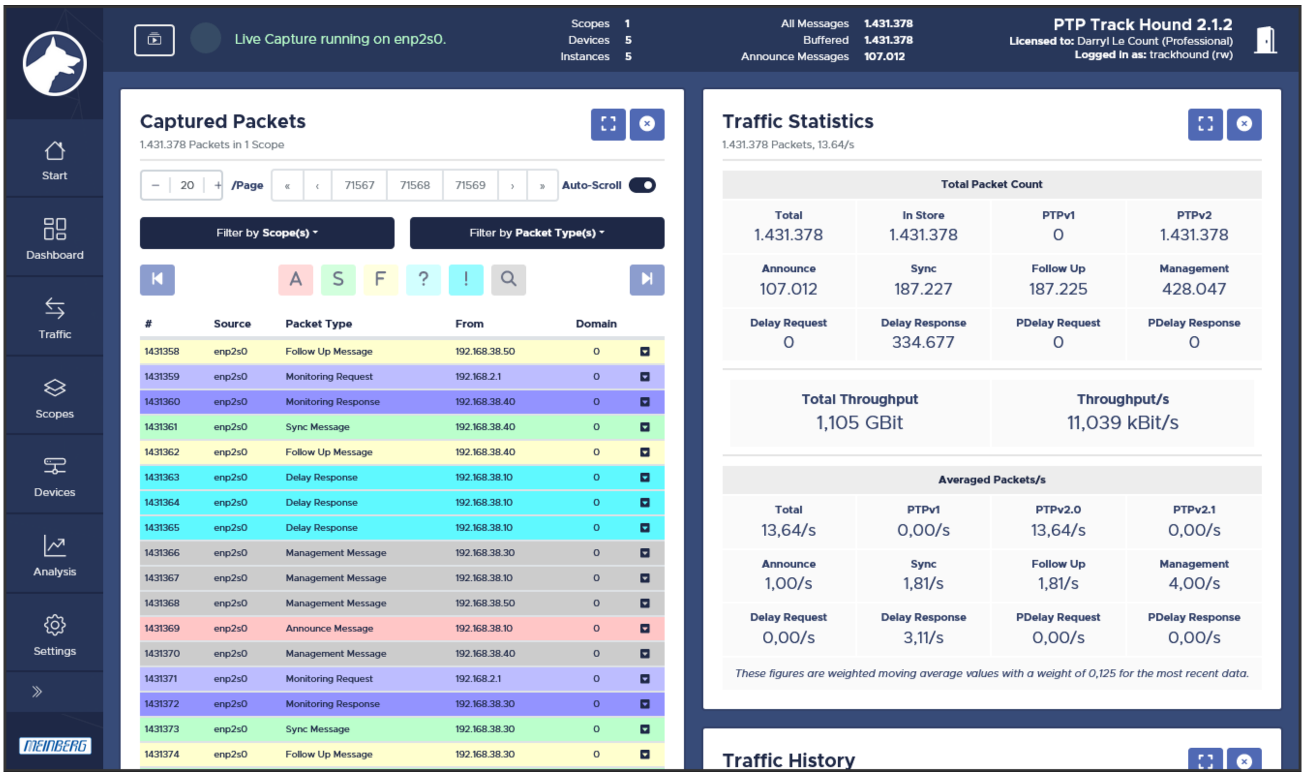
Tiles can also be removed from the Dashboard window directly by clicking on the "**Close Window**" button at the top right of each tile.

Tiles can also be expanded to fill the page by clicking on the "**Enter Fullscreen**" button.

For information on the tiles available in each of the categories, please consult:

- Chapter 8, "**Traffic**"
- Chapter 9, "**Scopes**"
- Chapter 10, "**Devices**"
- Chapter 11, "**Analysis**"

8 Traffic



Live Capture running on enp2s0.

Scopes 1
Devices 5
Instances 5

All Messages 1.431.378
Buffered 1.431.378
Announce Messages 107.012

PTP Track Hound 2.1.2
Licensed to: Darryl Le Count (Professional)
Logged in as: trackhound (rw)

Captured Packets
1.431.378 Packets in 1 Scope

– 20 + /Page « ‹ 71567 71568 71569 › »

#	Source	Packet Type	From	Domain
1431358	enp2s0	Follow Up Message	192.168.38.50	0
1431359	enp2s0	Monitoring Request	192.168.2.1	0
1431360	enp2s0	Monitoring Response	192.168.38.40	0
1431361	enp2s0	Sync Message	192.168.38.40	0
1431362	enp2s0	Follow Up Message	192.168.38.40	0
1431363	enp2s0	Delay Response	192.168.38.10	0
1431364	enp2s0	Delay Response	192.168.38.10	0
1431365	enp2s0	Delay Response	192.168.38.10	0
1431366	enp2s0	Management Message	192.168.38.30	0
1431367	enp2s0	Management Message	192.168.38.10	0
1431368	enp2s0	Management Message	192.168.38.50	0
1431369	enp2s0	Announce Message	192.168.38.10	0
1431370	enp2s0	Management Message	192.168.38.40	0
1431371	enp2s0	Monitoring Request	192.168.2.1	0
1431372	enp2s0	Monitoring Response	192.168.38.30	0
1431373	enp2s0	Sync Message	192.168.38.30	0
1431374	enp2s0	Follow Up Message	192.168.38.30	0

Traffic Statistics
1.431.378 Packets, 13.64/s

Total Packet Count			
Total 1.431.378	In Store 1.431.378	PTPv1 0	PTPv2 1.431.378
Announce 107.012	Sync 187.227	Follow Up 187.225	Management 428.047
Delay Request 0	Delay Response 334.677	PDelay Request 0	PDelay Response 0

Total Throughput	Throughput/s
1,105 GBit	11,039 kBit/s

Averaged Packets/s			
Total 13,64/s	PTPv1 0,00/s	PTPv2.0 13,64/s	PTPv2.1 0,00/s
Announce 1,00/s	Sync 1,81/s	Follow Up 1,81/s	Management 4,00/s
Delay Request 0,00/s	Delay Response 3,11/s	PDelay Request 0,00/s	PDelay Response 0,00/s

These figures are weighted moving average values with a weight of 0,125 for the most recent data.

Traffic History

Figure 8.1: Traffic

The **Traffic** section provides a detailed overview of the captured traffic and the results of the analyses performed on this traffic. It comprises three tiles: **Captured Packets**, **Traffic Statistics**, and **Traffic History**.

Captured Packets

The **Captured Packets** tile displays the most recent PTP messages captured by the capture service or, if a capture file is being analyzed, the contents of that capture file.

This tile can be expanded to encompass the entire browser window by clicking on the **Enter Fullscreen** button.

To hide this tile, click on the "**Close Window**" button. This button will then remain hidden from view on the Dashboard until your browser's Web Storage is cleared or the tile is reopened by clicking on "**Reset View**" at the bottom right of the page or selecting "**Captured Packets**" from the "**Closed Windows**" menu, also at the bottom right of the page.

The list is regularly refreshed to show the latest PTP messages; the list refresh can be suspended by disabling the **Auto-Scroll** toggle switch, which allows you to freely navigate between the different pages in the message list. While the **Auto-Scroll** toggle switch is enabled, only the most recent page will be displayed.

The number of messages displayed per page can be adjusted using the "**/Page**" +/- buttons at the top left of the tile.

The list can be filtered by selecting either "**Filter by Scope(s)**" (see Chapter 9, "**Scopes**") or "**Filter by Packet Type(s)**" and selecting the checkboxes for each of the scopes or packet types respectively that are to be displayed in the list. Filtering is possible even while **Auto-Scroll** is enabled. Active filters are displayed beneath the filter buttons and can be removed individually by clicking on the filter button or completely by clicking on "**Clear All**".

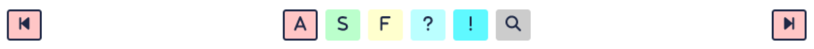


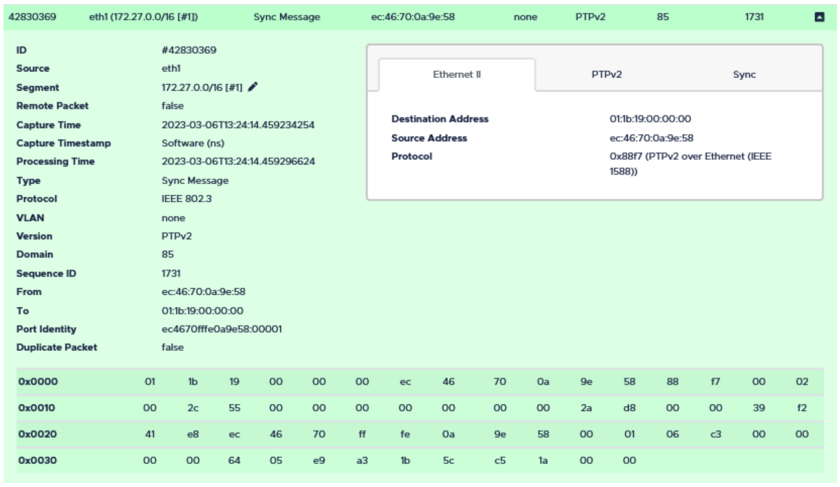
Figure 8.2: Packet Navigator Toolbar

It is also possible to use the Packet Navigator toolbar to quickly navigate between messages of the same type within the list, for which **Auto-Scroll** must be disabled:

- Clicking on the A button will jump to the next *Announce* message.
- Clicking on the S button will jump to the next *Sync* message.
- Clicking on the F button will jump to the next *Follow-Up* message.
- Clicking on the ? button will jump to the next *(Peer) Delay Request* message.
- Clicking on the ! button will jump to the next *(Peer) Delay Response* message.
- Clicking on the **magnifying glass** button will jump to the next *Management* message.

When one of these buttons is selected, the two buttons at the far ends of the Packet Navigator toolbar can be used to navigate to the previous or next message of the same time.

#	The ID number of the PTP message as assigned by PTP Track Hound v2.
Source	The interface through which the capture instance received the PTP message or, if analyzing a capture file, the file to which the message relates.
Type	The type of PTP message.
From	The IP address or MAC address from which the PTP message originated.
VLAN	If VLAN tagging is used, this shows the VLAN tag under which the PTP message has been sent.
Version	The PTP version for this message.
Domain	For PTPv2 or PTPv2.1 messages, this will show the PTP domain number. For PTPv1 messages, this will show the PTP subdomain.
Sequence ID	The sequential ID number of the PTP message as assigned by the PTP instance.



42830369	eth1 (172.27.0.0/16 [#1])	Sync Message	ec:46:70:0a:9e:58	none	PTPv2	85	1731

ID	#42830369
Source	eth1
Segment	172.27.0.0/16 [#1]
Remote Packet	false
Capture Time	2023-03-06T13:24:14.459234254
Capture Timestamp	Software (ns)
Processing Time	2023-03-06T13:24:14.459296624
Type	Sync Message
Protocol	IEEE 802.3
VLAN	none
Version	PTPv2
Domain	85
Sequence ID	1731
From	ec:46:70:0a:9e:58
To	01:1b:19:00:00:00
Port Identity	ec4670fffe0a9e58:00001
Duplicate Packet	false

Destination Address	01:1b:19:00:00:00
Source Address	ec:46:70:0a:9e:58
Protocol	0x88f7 (PTPv2 over Ethernet (IEEE 1588))

```
0x0000  01 1b 19 00 00 00 ec 46 70 0a 9e 58 88 f7 00 02
0x0010  00 2c 55 00 00 00 00 00 00 00 2a d8 00 00 39 f2
0x0020  41 e8 ec 46 70 ff fe 0a 9e 58 00 01 06 c3 00 00
0x0030  00 00 64 05 e9 a3 1b 5c c5 1a 00 00
```

Figure 8.3: PTP Message Breakdown

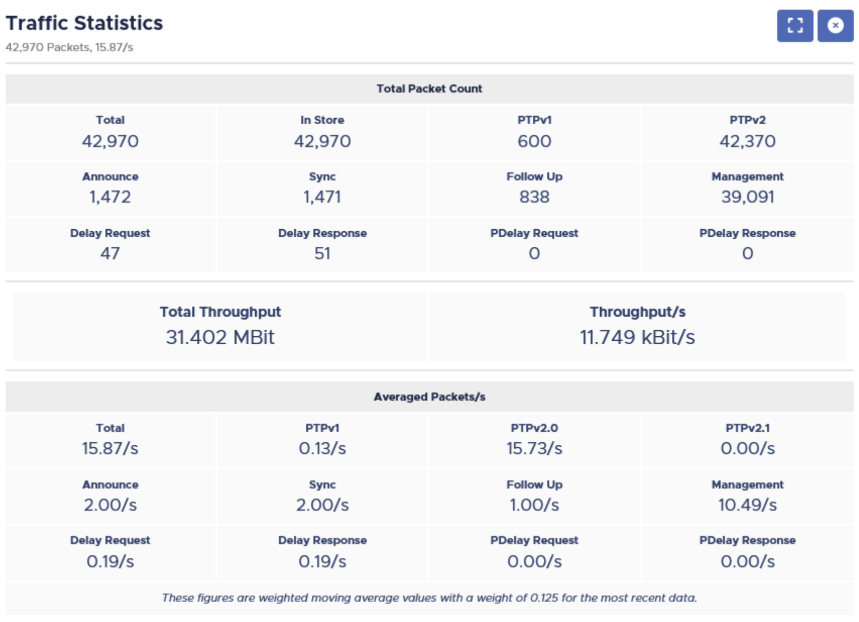
A breakdown of any given message can also be displayed by clicking on the **Expand** button of the message line. If **Auto-Scroll** is enabled, this will disable it. Expanding a PTP message in this way provides the following information in addition to the fields listed above:

Remote Packet	This indicates whether the message was captured on a remote PTP Track Hound v2 instance (*true*) or locally on this PTP Track Hound v2 (*false*).
Capture Time	The time at which the PTP Track Hound v2 instance captured the PTP message. The time in this case is based on the local clock of the device on which PTP Track Hound v2 captured the data.
Capture Timestamp	Specifies at which level the capture timestamp was taken for this message: *hardware* (hardware timestamper in network controller) or *software*. Also shows the precision of the timestamp (microseconds μs or nanoseconds *ns*).
Processing Time	The time at which the PTP Track Hound v2 instance processed and analyzed the PTP message. This provides an indication of the time gap between capture and processing, which will be accordingly shorter on more powerful systems. When forwarding capture data from one instance to another remote instance, this is also indicative of network delays. The time in this case is based on the local clock of the device on which PTP Track Hound v2 processed the data.
Protocol	The network protocol over which this message was transmitted and received (*IPv4*, *IPv6* or *IEEE802.3*).
To	For multicast PTP traffic, this will be the multicast address. For unicast PTP traffic captured via a dedicated monitoring interface (port mirroring), this will be the unicast address.
Port Identity	The PTP clock port identity.
Duplicate Packet	Specifies whether the captured message is a duplicate of another (for example, in a PRP network).

The white panel within the table provides a layer-by-layer breakdown of the contents of the PTP message as applicable, with the tabs from left to right showing the message data at the Ethernet level (Layer 2), IPv4/IPv6 level (Layer 3, if message is received via an IP network), transport level (UDP, Layer 4, if message is received via an IP network), and PTP application level (Layer 5) respectively. If the PTP message contains TLV data, that too will also be displayed in separate tabs.

The table below the list of fields shows a byte-by-byte breakdown of the message.

Traffic Statistics



Traffic Statistics
42,970 Packets, 15.87/s

Total Packet Count

Total 42,970	In Store 42,970	PTPv1 600	PTPv2 42,370
Announce 1,472	Sync 1,471	Follow Up 838	Management 39,091
Delay Request 47	Delay Response 51	PDelay Request 0	PDelay Response 0

Total Throughput	Throughput/s
31.402 MBit	11.749 kBit/s

Averaged Packets/s

Total 15.87/s	PTPv1 0.13/s	PTPv2.0 15.73/s	PTPv2.1 0.00/s
Announce 2.00/s	Sync 2.00/s	Follow Up 1.00/s	Management 10.49/s
Delay Request 0.19/s	Delay Response 0.19/s	PDelay Request 0.00/s	PDelay Response 0.00/s

These figures are weighted moving average values with a weight of 0.125 for the most recent data.

Figure 8.4: Traffic Statistics

The **Traffic Statistics** tile displays general statistics on PTP throughput.

This window can be expanded to encompass the entire browser window by clicking on the **Enter Fullscreen** button.

To hide this tile, click on the "**Close Window**" button. This button will then remain hidden from view on the Dashboard until your browser's Web Storage is cleared or the tile is reopened by clicking on "**Reset View**" at the bottom right of the page or selecting "**Traffic Statistics**" from the "**Closed Windows**" menu, also at the bottom right of the page.

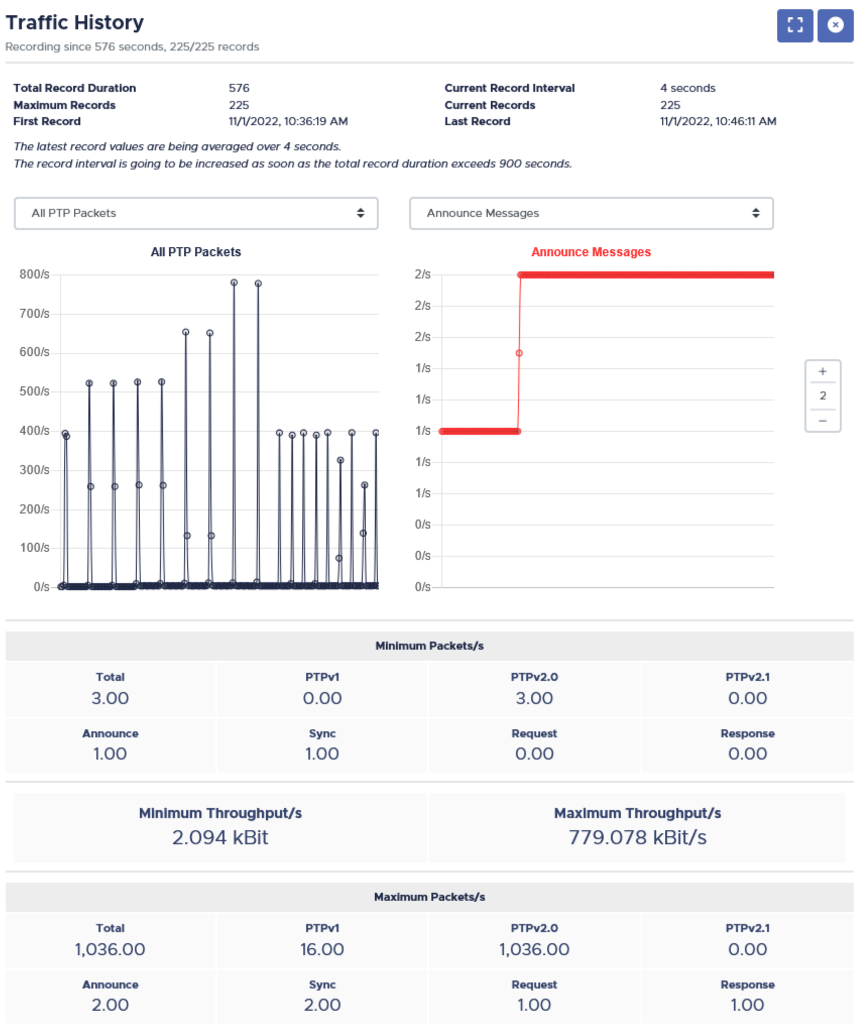
The **Total Packet Count** section provides the following values:

Total	This is the total number of incoming PTP messages counted since last erasure.
In Store	This is the total number of PTP messages that are held in the current capture log. As long as the memory buffer has not been filled, this will generally be equal to the value **Total**, but if the memory buffer becomes full, older PTP messages will be purged from the capture log and this value will diverge accordingly.
PTPv1	The number of incoming PTPv1 messages counted since last erasure.
PTPv2	The number of incoming PTPv2 and PTPv2.1 messages counted since last erasure.
Announce	The number of Announce messages currently recorded in the capture log.
Sync	The number of Sync messages currently recorded in the capture log.
Follow Up	The number of Follow Up messages currently recorded in the capture log.
Management	The number of Management messages currently recorded in the capture log.
Delay Request	The number of Delay Request messages for E2E measurement currently recorded in the capture log.
Delay Response	The number of Delay Response messages for E2E measurement currently recorded in the capture log.
PDelay Request	The number of Delay Request messages for P2P measurement currently recorded in the capture log.
PDelay Response	The number of Delay Response messages for P2P measurement currently recorded in the capture log.

Corresponding rate measurements for the numbers of each of these message types captured per second are shown at the bottom of the tile. These figures are weighted rolling average values with the most recent data weighted at 0.125.

The **Total Throughput** and **Throughput/s** values shown in the middle of the tile represent the total PTP traffic volume and PTP traffic data rate per second respectively. The **Throughput/s** value in particular can be especially useful for ensuring that PTP measurements are not distorted by a traffic bottleneck at any point in the network.

Traffic History



Traffic History
Recording since 576 seconds, 225/225 records

Total Record Duration	576	**Current Record Interval**	4 seconds
Maximum Records	225	**Current Records**	225
First Record	11/1/2022, 10:36:19 AM	**Last Record**	11/1/2022, 10:46:11 AM

The latest record values are being averaged over 4 seconds.
The record interval is going to be increased as soon as the total record duration exceeds 900 seconds.

All PTP Packets | Announce Messages

Minimum Packets/s

Total	PTPv1	PTPv2.0	PTPv2.1
3.00	0.00	3.00	0.00
Announce	**Sync**	**Request**	**Response**
1.00	1.00	0.00	0.00

Minimum Throughput/s	Maximum Throughput/s
2.094 kBit	779.078 kBit/s

Maximum Packets/s

Total	PTPv1	PTPv2.0	PTPv2.1
1,036.00	16.00	1,036.00	0.00
Announce	**Sync**	**Request**	**Response**
2.00	2.00	1.00	1.00

Figure 8.5: Traffic History

This tile provides a number of graphs that indicate the level of PTP traffic passing through the capture interfaces over time.

This tile can be expanded to encompass the entire browser window by clicking on the **Enter Fullscreen** button.

To hide this tile, click on the "**Close Window**" button. This button will then remain hidden from view on the Dashboard until your browser's Web Storage is cleared or the tile is reopened by clicking on "**Reset View**" at the bottom right of the page or selecting "**Traffic History**" from the "**Closed Windows**" menu, also at the bottom right of the page.

By default, two graphs are displayed in this tile and each graph can be configured to display a specific traffic history value by selecting it in each graph's drop-down menu. To add more graphs or remove existing ones, use the numerical value selector on the right side of the tile. Click on + to add a graph, and - to remove the last graph in the tile.

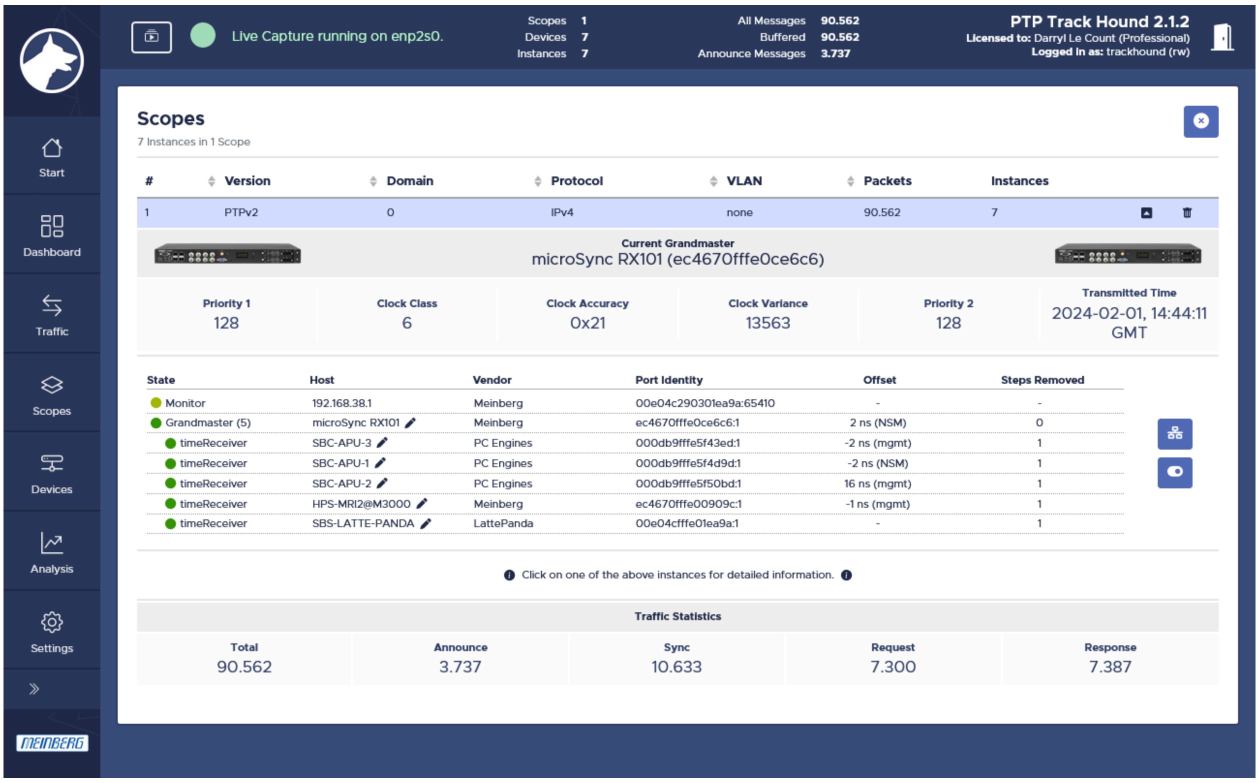
The measurement interval is dynamically adjusted to account for the amount of data stored; as the data volume increases, measurements are taken at less frequent intervals.

Any given data point on a graph can be reviewed by hovering the mouse cursor over it; this will display the measured value and the corresponding time and date.

Preferences for traffic history graphs are stored in your browser's Web Storage; erasing your browser's Web Storage data will therefore reset these graph preferences to the default (two graphs entitled **All PTP Packets** and **Announce Messages**).

The statistics displayed beneath the graphs show the minimum and maximum statistical values for each PTP message type in the same way as the traffic statistics above.

9 Scopes



Live Capture running on enp2s0.

Scopes	1
Devices	7
Instances	7

All Messages	90.562
Buffered	90.562
Announce Messages	3.737

PTP Track Hound 2.1.2
Licensed to: Darryl Le Count (Professional)
Logged in as: trackhound (rw)

Scopes
7 Instances in 1 Scope

#	Version	Domain	Protocol	VLAN	Packets	Instances
1	PTPv2	0	IPv4	none	90.562	7

Current Grandmaster
microSync RX101 (ec4670fffe0ce6c6)

Priority 1	Clock Class	Clock Accuracy	Clock Variance	Priority 2	Transmitted Time
128	6	0x21	13563	128	2024-02-01, 14:44:11 GMT

State	Host	Vendor	Port Identity	Offset	Steps Removed
● Monitor	192.168.38.1	Meinberg	00e04c290301ea9a:65410	-	-
● Grandmaster (5)	microSync RX101	Meinberg	ec4670fffe0ce6c6:1	2 ns (NSM)	0
● timeReceiver	SBC-APU-3	PC Engines	000db9fffe5f43ed:1	-2 ns (mgmt)	1
● timeReceiver	SBC-APU-1	PC Engines	000db9fffe5f4d9d:1	-2 ns (NSM)	1
● timeReceiver	SBC-APU-2	PC Engines	000db9fffe5f50bd:1	16 ns (mgmt)	1
● timeReceiver	HPS-MRI2@M3000	Meinberg	ec4670fffe00909c:1	-1 ns (mgmt)	1
● timeReceiver	SBS-LATTE-PANDA	LattePanda	00e04cfffe01ea9a:1	-	1

Click on one of the above instances for detailed information.

Traffic Statistics

Total	Announce	Sync	Request	Response
90.562	3.737	10.633	7.300	7.387

Figure 9.1: Scopes

Scopes are a feature specific to PTP Track Hound v2 that allow for meaningful analysis of relationships between PTP clocks. It allows multiple PTP networks existing within a single Ethernet or IP network to be visualized individually without having to manually identify which clock serves which purpose.

A scope is defined as a group of clocks that are capable of serving and receiving time to and from one another. This means that all clocks within a scope must share a common network protocol, PTP version (PTPv2 and PTPv2.1 are treated as a single version), PTP domain or subdomain, VLAN tag, and Grandmaster clock, and are assumed to be able to reach one another within a shared network or subnet.

It is also possible to manually assign instances to different segments according to the network interface or remote instance through which the PTP traffic is captured. Segmentation must be defined manually because a single instance of PTP Track Hound v2 has no way of reliably determining how external clocks in discrete subnets are able to communicate with one another.

Refer to Chapter 12, "Settings" for more information on segmentation.

Scopes are best illustrated using three concrete examples:

Example 1

A single network in which all devices can reach each other has 30 PTP instances, all running PTPv2 over IPv4 on PTP domain number *34*, and no VLAN tag.

PTP Track Hound v2 will detect all 30 PTP instances as belonging to one single scope.

Example 2

A single network in which all devices can reach each other has 20 devices only running a PTPv2 instance each over IPv4 on PTP domain number *34* and no VLAN tag. It also has another nine devices, each of which is only running a PTPv1 instance over IPv4 on PTP subdomain *_DFLT*. There is one further device that acts as a PTP bridge, running both a PTPv2 instance on domain *34* and a PTPv1 instance on subdomain *_DFLT*.

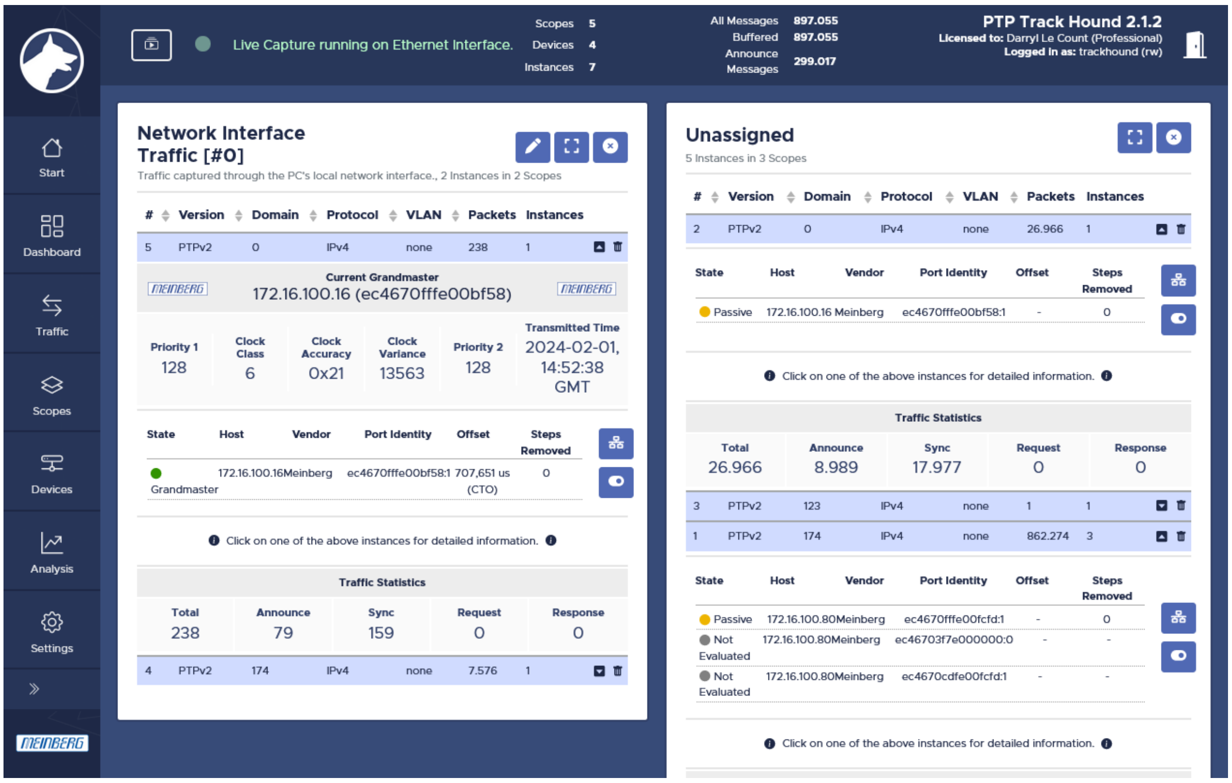
PTP Track Hound v2 in this case will detect two different scopes. The first scope will contain all of the PTPv2 instances, while the second scope will contain all of the PTPv1 instances. The PTP bridge running both PTPv2 and PTPv1 instances will be present in both scopes.

Example 3

A network has two subnets, both of which are reachable from the device running PTP Track Hound v2 via two different Ethernet interfaces. Each of these subnets has ten devices running a PTPv2.1 instance each, over IPv4 on PTP domain number *60* and no VLAN tag. The traffic from ten of these devices is captured via Ethernet Interface 1 on the device running PTP Track Hound v2; that interface is assigned the segment value **1**. The traffic from the other ten devices is captured via Ethernet Interface 2 on the very same device running PTP Track Hound v2; that interface is assigned the segment value **2**.

With this segmentation, PTP Track Hound v2 will detect two different scopes and will represent these as two different networks. Without this segmentation, PTP Track Hound v2 will detect all instances as belonging to a single scope, even though it is unknown if clocks in different subnets can reach one another.

Segmentation



Live Capture running on Ethernet Interface.

Scopes	5
Devices	4
Instances	7
All Messages	897.055
Buffered	897.055
Announce Messages	299.017

PTP Track Hound 2.1.2
Licensed to: Darryl Le Count (Professional)
Logged in as: trackhound (rw)

Network Interface Traffic [#0]
Traffic captured through the PC's local network interface., 2 Instances in 2 Scopes

#	Version	Domain	Protocol	VLAN	Packets	Instances
5	PTPv2	0	IPv4	none	238	1

Current Grandmaster
172.16.100.16 (ec4670fffe00bf58)

Priority 1	Clock Class	Clock Accuracy	Clock Variance	Priority 2	Transmitted Time
128	6	0x21	13563	128	2024-02-01, 14:52:38 GMT

State	Host	Vendor	Port Identity	Offset	Steps Removed
● Grandmaster	172.16.100.16	Meinberg	ec4670fffe00bf58:1	707,651 us (CTO)	0

Click on one of the above instances for detailed information.

Traffic Statistics

Total	Announce	Sync	Request	Response
238	79	159	0	0

#	Version	Domain	Protocol	VLAN	Packets	Instances
4	PTPv2	174	IPv4	none	7.576	1

Unassigned
5 Instances in 3 Scopes

#	Version	Domain	Protocol	VLAN	Packets	Instances
2	PTPv2	0	IPv4	none	26.966	1

State	Host	Vendor	Port Identity	Offset	Steps Removed
● Passive	172.16.100.16	Meinberg	ec4670fffe00bf58:1	-	0

Click on one of the above instances for detailed information.

Traffic Statistics

Total	Announce	Sync	Request	Response
26.966	8.989	17.977	0	0

#	Version	Domain	Protocol	VLAN	Packets	Instances
3	PTPv2	123	IPv4	none	1	1
1	PTPv2	174	IPv4	none	862.274	3

State	Host	Vendor	Port Identity	Offset	Steps Removed
● Passive	172.16.100.80	Meinberg	ec4670fffe00fcfd:1	-	0
● Not Evaluated	172.16.100.80	Meinberg	ec46703f7e000000:0	-	-
● Not Evaluated	172.16.100.80	Meinberg	ec4670cdfe00fcfd:1	-	-

Click on one of the above instances for detailed information.

Figure 9.2: Scopes Broken Down into Segments

If none of the capture data currently held in the capture log is assigned to a specific segment ID, the **Scopes** section will consist of a single tile showing all scopes automatically detected by PTP Track Hound v2.

If there is capture data held in the capture log that is assigned to specific segment IDs, the **Scopes** section will consist of multiple tiles, one for each of the manually defined network segments.

Please note that segments are based on the captured data currently stored in the capture log; even if all segment IDs are removed under Settings, the segments for the previously captured data will remain displayed on the **Scopes** page until the capture log is erased. Conversely, if individual traffic sources are assigned to specific segment IDs while a capture session is running, the data captured previously will remain 'unassigned'. Segment IDs should therefore ideally be set and the existing capture log should be backed up and erased from PTP Track Hound v2 as necessary before a new capture session is started using new segment IDs.

The name and description of each segment ID can be modified by clicking on the **Edit** icon at the top right of each segment tile.

This name and description will then be displayed at the top of each tile as appropriate. Please note that the name and description are bound to that segment ID; changing the segment ID of a traffic source will not carry the name and description over with it.

Layout of Scopes Section

The list of detected scopes is shown at the top of the tile, showing the common PTP version, (sub)domain, network protocol, and VLAN tag of the instances in each scope. This list also shows the number of captured packets attributable to that scope as well as the number of instances within that scope.

Clicking on a scope will expand the scope details to show details on the current Grandmaster of that scope, including the criteria used by the Best Master Clock Algorithm for establishing that clock's suitability as the Grandmaster.

Beneath this is the list of instances receiving time from the Grandmaster, showing the relative offset to the Grandmaster and the number of steps removed between the receiving instance and the Grandmaster.

This list can also be shown as an organizational chart that graphically represents the topology of the clock hierarchy by clicking on the button "**Show Topology Graph**".

You can switch back to the tree list view at any time by clicking on the button "**Show Tree View**".

By default, inactive instances (instances that had been previously detected but are no longer reachable) are displayed in these views. If you wish to hide these, disable the toggle switch "**Include Inactive Instances**" to the right of the tree list view or topology view.

Selecting any instance within the tree list or topology view will display the metadata stored about that instance, its clock quality dataset, and traffic statistics for that instance. This information can be hidden again by clicking on the button "**Hide Details**".

The metadata for any given instance can be edited by clicking on the **Edit** icon next to the vendor or address/hostname in the detailed view.

Scope-specific statistics on PTP message types (*Announce*, *Sync*, *Delay Request*, *Delay Response*) are provided at the end of each scope section.

Information:

The information previously revealed by the "**Show Statistics**" button under PTP Track Hound v2.0.x has been relocated to the **Analysis** section of the PTP Track Hound v2 user interface.

Please refer to Chapter 11, "Analysis" for more information.

Illegal Grandmasters

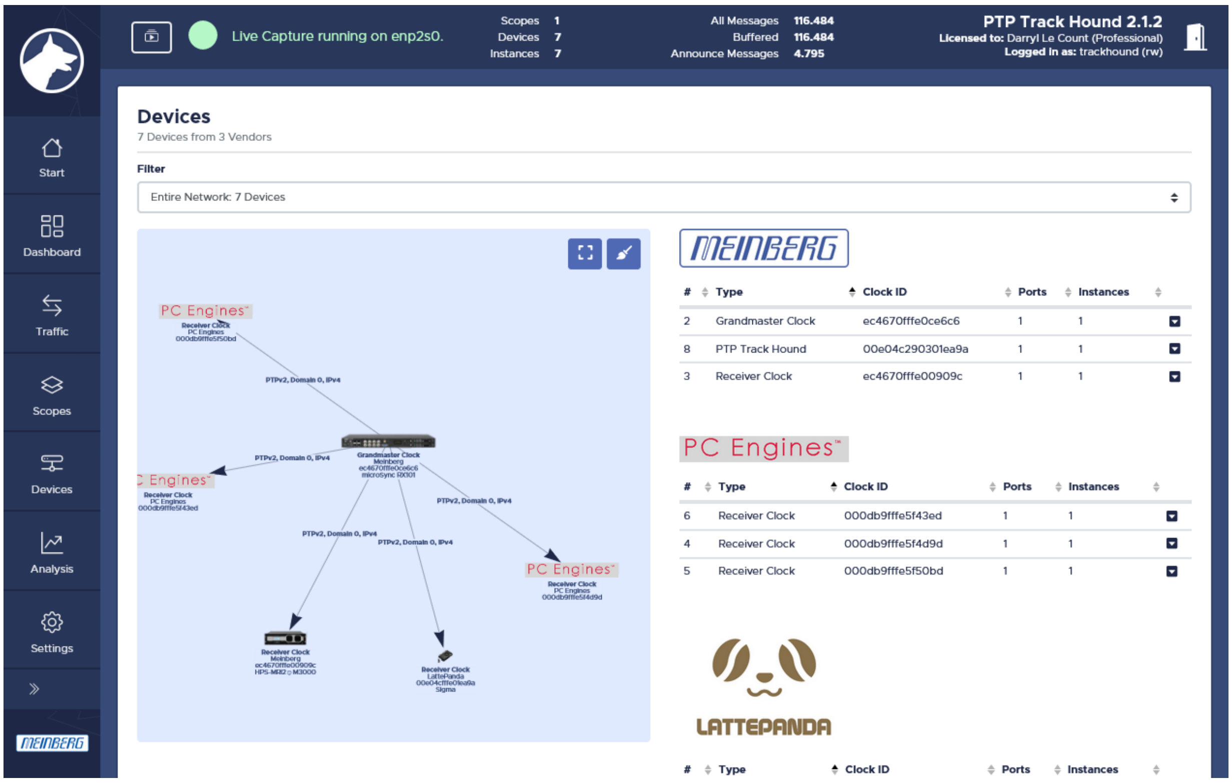
In certain network configurations you may observe a single scope having multiple Grandmasters, of which only one (the clock with the best Clock Class) is declared to be the actual Grandmaster while the others are deemed to be 'illegal'.

These 'illegal' Grandmasters will be displayed in the Scopes section with a prominent pink background and are also shown in red text in the Devices section.

There are two possible reasons why this may occur:

- If you have set your PTP Track Hound v2 instance up to monitor PTP clocks that have identical protocols, PTP versions, sub(domains), and VLAN tags but are split into multiple subnets, PTP Track Hound v2 will by default not differentiate between these subnets. Thus, while these are effectively two separate PTP networks, PTP Track Hound v2 recognizes them by default as a single network. In this case, you should use separate network interfaces (or remote instances) to capture traffic from each of these subnets and assign unique segment IDs to these segments to represent the different subnets.
- The presence of illegal Grandmasters within a single network may also be indicative of unintentional network problems, whereby PTP Track Hound v2 is able to reach multiple PTP clocks but some of these clocks have no functioning path to each other. In this case, a review of your network infrastructure is recommended to identify whether there may be PTP islanding (for example, caused by failure of a boundary clock).

10 Devices



Live Capture running on enp2s0.

Scopes 1
Devices 7
Instances 7

All Messages 116.484
Buffered 116.484
Announce Messages 4.795

PTP Track Hound 2.1.2
Licensed to: Darryl Le Count (Professional)
Logged in as: trackhound (rw)

Devices
7 Devices from 3 Vendors

Filter
Entire Network: 7 Devices

MEINBERG

#	Type	Clock ID	Ports	Instances
2	Grandmaster Clock	ec4670fffe0ce6c6	1	1
8	PTP Track Hound	00e04c290301ea9a	1	1
3	Receiver Clock	ec4670fffe00909c	1	1

PC Engines

#	Type	Clock ID	Ports	Instances
6	Receiver Clock	000db9fffe5f43ed	1	1
4	Receiver Clock	000db9fffe5f4d9d	1	1
5	Receiver Clock	000db9fffe5f50bd	1	1

LATTEPANDA

#	Type	Clock ID	Ports	Instances

Figure 10.1: Devices

The **Devices** section provides a detailed overview of the detected devices and their associated instances. It is divided into three sections:

Filter

By default, the **Devices** section shows all detected devices. The filter drop-down menu can be used to have the map and list show only clocks that have a known group relationship with other clocks. These groups of devices are roughly analogous to scopes and their instances. It is also possible to only show 'detached' devices, that is, clocks with no known relationship to any other clock and are considered to be effectively 'orphaned'.

Map

The virtual device map shows a representation of the instances and their relationships with other device. Arrows are used to represent the relationship between each pair of devices, with the arrow pointing towards the device hosting an instance that is receiving time. If these relationships are defined by a manually defined segment, these arrows will be annotated with the defining features of that relationship, specifically the segment name, their PTP version, domain or subdomain, and network protocol.

Clicking on the vendor logo of any device on the map will expand the device in the list to the right, showing the instances running on that device.

Individual devices can be moved around within the map by dragging them to the desired location.

The view can be zoomed and centered automatically on the clocks on the map by clicking on the "**Tidy Up**" button at the top right of the map. Please note, however, that this view is not preserved and will be lost as soon as the page is closed.

A clock shown with red text is an 'illegal' Grandmaster; refer to Chapter 9, "**Scopes**" for more information.

Information:

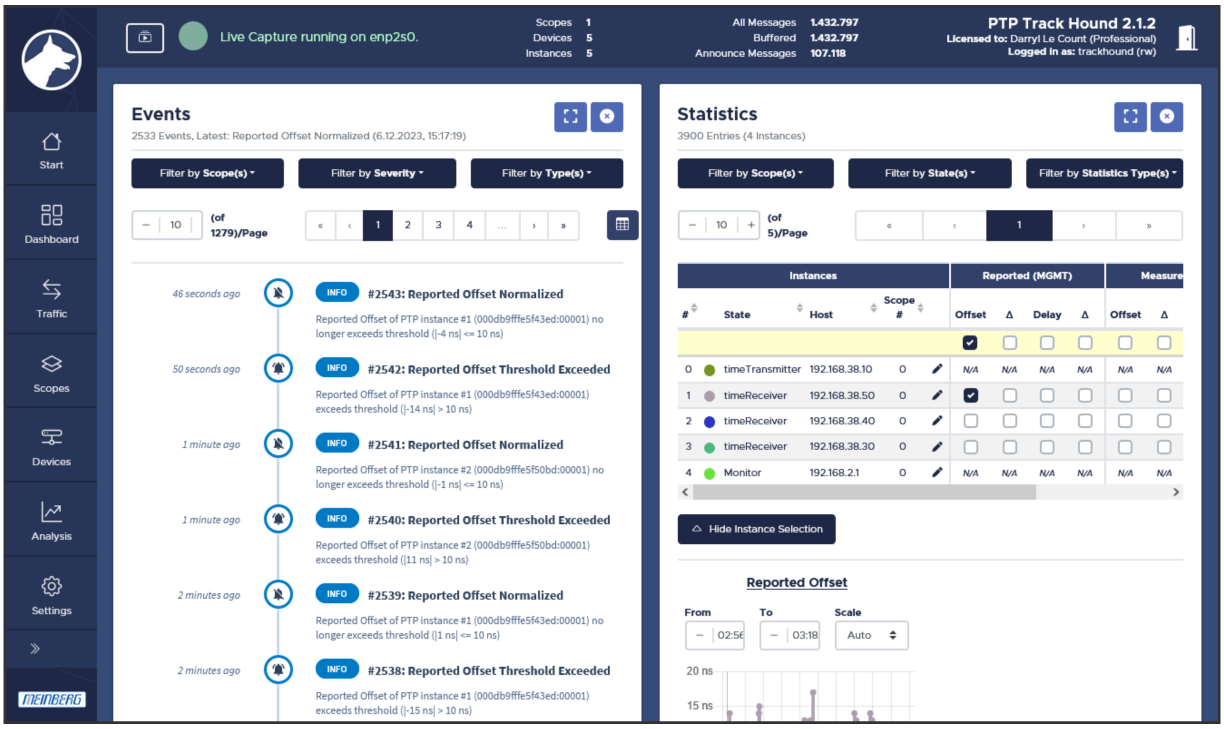
If PTP Track Hound v2 has been configured to transmit PTP management messages, it too will be shown as a PTP device in the virtual device map and device list.

List

Each of the devices is shown to the right of the map, grouped by vendor. Clicking on any device will expand the device, showing the instances running on that device; this will also highlight the corresponding device on the map in a slightly larger font.

The metadata for each device can be manually edited by selecting it either in the list or virtual device map and clicking on the "**Customize**" pencil icon to the right.

11 Analysis



Live Capture running on enp2s0.

Scopes 1
Devices 5
Instances 5

All Messages 1.432.797
Buffered 1.432.797
Announce Messages 107.118

PTP Track Hound 2.1.2
Licensed to: Darryl Le Count (Professional)
Logged in as: trackhound (rw)

Events

2533 Events, Latest: Reported Offset Normalized (6.12.2023, 15:17:19)

10 (of 1279)/Page « ‹ 1 2 3 4 … › »

46 seconds ago INFO **#2543: Reported Offset Normalized**
Reported Offset of PTP instance #1 (000db9fffe5f43ed:00001) no longer exceeds threshold (|-4 ns| <= 10 ns)

50 seconds ago INFO **#2542: Reported Offset Threshold Exceeded**
Reported Offset of PTP instance #1 (000db9fffe5f43ed:00001) exceeds threshold (|-14 ns| > 10 ns)

1 minute ago INFO **#2541: Reported Offset Normalized**
Reported Offset of PTP instance #2 (000db9fffe5f50bd:00001) no longer exceeds threshold (|-1 ns| <= 10 ns)

1 minute ago INFO **#2540: Reported Offset Threshold Exceeded**
Reported Offset of PTP instance #2 (000db9fffe5f50bd:00001) exceeds threshold (|11 ns| > 10 ns)

2 minutes ago INFO **#2539: Reported Offset Normalized**
Reported Offset of PTP instance #1 (000db9fffe5f43ed:00001) no longer exceeds threshold (|1 ns| <= 10 ns)

2 minutes ago INFO **#2538: Reported Offset Threshold Exceeded**
Reported Offset of PTP instance #1 (000db9fffe5f43ed:00001) exceeds threshold (|-15 ns| > 10 ns)

Statistics

3900 Entries (4 Instances)

10 (of 5)/Page « ‹ 1 › »

Instances					Reported (MGMT)				Measure	
#		State	Host	Scope #	Offset	Δ	Delay	Δ	Offset	Δ
					✓	☐	☐	☐	☐	☐
0	●	timeTransmitter	192.168.38.10	0	N/A	N/A	N/A	N/A	N/A	N/A
1	●	timeReceiver	192.168.38.50	0	✓	☐	☐	☐	☐	☐
2	●	timeReceiver	192.168.38.40	0	☐	☐	☐	☐	☐	☐
3	●	timeReceiver	192.168.38.30	0	☐	☐	☐	☐	☐	☐
4	●	Monitor	192.168.2.1	0	N/A	N/A	N/A	N/A	N/A	N/A

Reported Offset

From 02:56 To 03:18 Scale Auto

Figure 11.1: Analysis

The **Analysis** page is a new feature of the PTP Track Hound v2 user interface since v2.1.0 and accommodates much of the analysis-related content that was previously located on the **Scopes** page and the **Dashboard** in v2.0.x.

Events

The **Events** tile shows a timeline of relevant events related to the PTP messages captured by the capture service, starting from the most recent event and progressing backwards over time. This list can be filtered as necessary to limit the output to information relating to specific scopes, severity levels, or event types.

This tile can be expanded to encompass the entire browser window by clicking on the **Enter Fullscreen** button.

To hide this tile, click on the "**Close Window**" button. This tile will then remain hidden from view on the Dashboard until your browser's Web Storage is cleared or the tile is reopened by clicking on "**Reset View**" at the bottom right of the page or selecting "**Events**" from the "**Closed Windows**" menu, also at the bottom right of the page.

The list is regularly refreshed to show the latest events.

The number of messages displayed per page can be adjusted using the "**/Page**" +/- buttons at the top left of the tile.

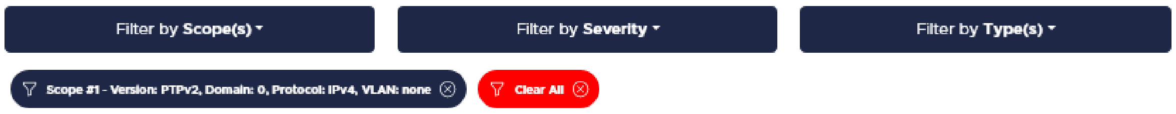
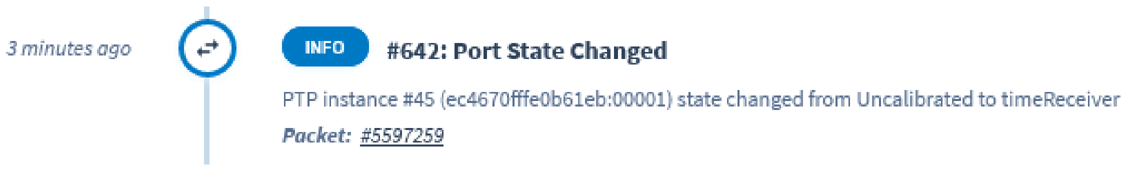

Filter by Scope(s)
Filter by Severity
Filter by Type(s)
Scope #1 - Version: PTPv2, Domain: 0, Protocol: IPv4, VLAN: none
Clear All

Figure 11.2: Event Filter Toolbar

The list can be filtered by selecting either "**Filter by Scope(s)**" (see Chapter 9, "Scopes"), "**Filter by Severity**", or "**Filter by Type(s)**" and selecting the checkboxes for each of the scopes, severity levels, and event types respectively that are to be displayed in the list. Active filters are displayed beneath the filter buttons and can be removed individually by clicking on the filter button or completely by clicking on "**Clear All**".

This tile can be switched to the classic "Event Table" view by clicking on the "**Show Event Table**" button and back again to the "Timeline" view by clicking on the "**Show Timeline**" button.



3 minutes ago

INFO **#642: Port State Changed**

PTP instance #45 (ec4670fffe0b61eb:00001) state changed from Uncalibrated to timeReceiver

Packet: #5597259

Figure 11.3: Example of an Event in the Timeline View

Regardless of whether the Timeline or Event Table view is selected, each event contains the same information:

- The amount of time that has passed since each event is shown on the far left of the Timeline view. Alternatively, the timestamp of the event is shown under "**Time (UTC)**" in the Event Table view.
- The color of the severity bubble and the icon (or the row background color in the Event Table view) indicates the severity of the event.
- The headline of each event contains the event ID and event type.
- The text beneath the headline shows a more detailed description of the event, including the clocks involved.
- If the event relates to a specific PTP message, the packet number is displayed beneath the description (or under the "**Packet**" column in the Event Table view. This can be clicked on to expand the packet details in the "Traffic" view.

Statistics

The **Statistics** tile provides statistical analyses based on the three analysis methods employed by PTP Track Hound v2 for each detected instance.

Each instance is listed in the table with the color used for graphs, the IP address or hostname, and the scope ID to which it has been assigned.

Each instance can have a custom name assigned to it and the color used to represent the instance in graphs can be changed by clicking on the "**Edit**" button.

The number of instances displayed per page can be adjusted using the "**/Page**" +/- buttons at the top left of the tile.

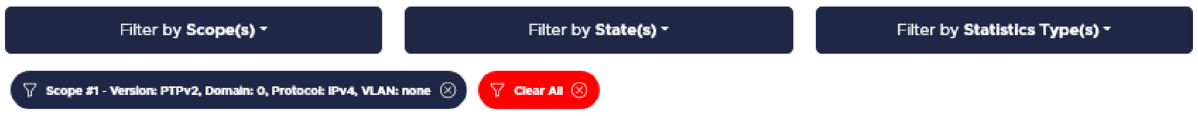

Filter by **Scope(s)**
Filter by **State(s)**
Filter by **Statistics Type(s)**
Scope #1 - Version: PTPv2, Domain: 0, Protocol: IPv4, VLAN: none
Clear All

Figure 11.4: Statistics Filter Toolbar

The list can be filtered by selecting either "**Filter by Scope(s)**" (see Chapter 9, "Scopes"), "**Filter by State(s)**", or "**Filter by Statistics Type(s)**" and selecting the checkboxes for each of the scopes, port states, and the *availability* of specific statistics types respectively that are to be displayed in the list. Active filters are displayed beneath the filter buttons and can be removed individually by clicking on the filter button or completely by clicking on "**Clear All**".

The table contains three main columns, representing each of the analysis methods: **Reported (MGMT)**, **Measured (NSM)**, and **CTO**. For more information on each analysis method, please refer to Chapter 14.2, "Types of Analysis in PTP Track Hound v2".

Under each main column, there are specific types of statistics:

- The "*Offset*" represents the absolute difference between the time of a timeReceiver and that of its timeTransmitter.
- The "*Delta*" (Δ) column after the *Offset* column represents the changes in offset between one measurement and the next and is indicative of offset stability or jitter (lower delta = lower jitter).
- The *Delay* represents the path delay between a timeReceiver and a timeTransmitter.
- The "*Delta*" (Δ) column after the *Delay* column represents changes in path delay between one measurement and the next and is indicative of offset stability or jitter (lower delta = lower jitter).

Spikes in offset delta or delay delta values can be indicative of sudden changes (whether deliberate or unexpected) in network conditions.

The presence of a checkbox in the table indicates that there is data available for that instance. *N/A* indicates that there is no data available, which is perfectly normal in most cases (timeTransmitters, for example, can of course not provide any offset or path delay data).

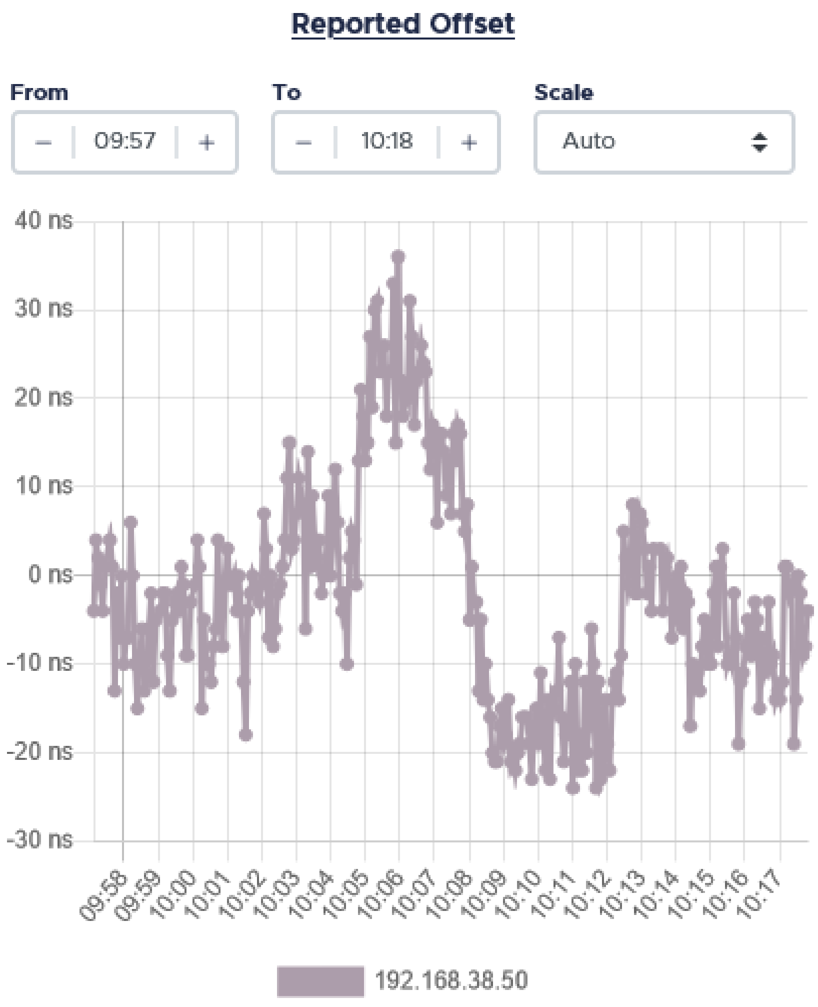


Figure 11.5: Graph based on Available Data

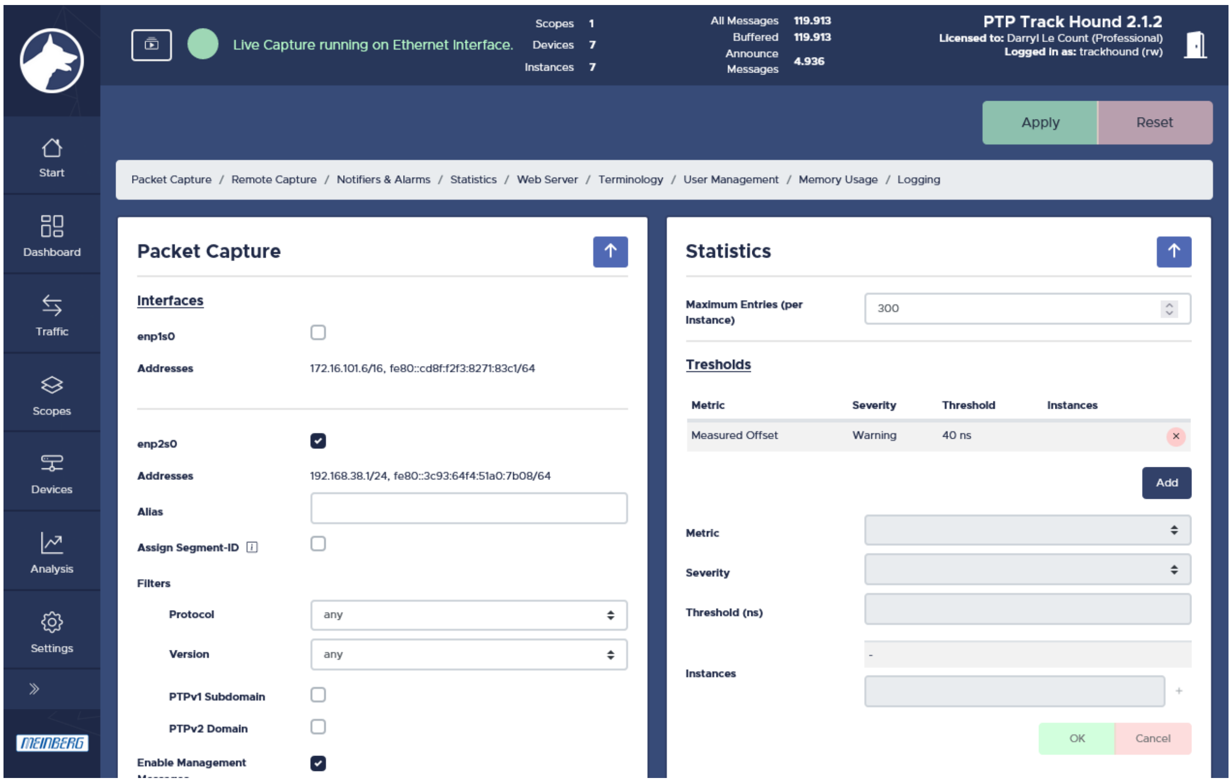
Selecting one of these checkboxes will display a graph for that dataset in the indicated color.

The scale of the X-axis of the graph initially encompasses the full time range available of the selected datasets, and will automatically expand as the available data grows, but can be limited to a specific time range using the **From** and **To** fields above each graph.

The scale of the Y-axis will automatically adapt to accommodate the spread of the values in the dataset, but can also be adapted to a range of scales available in the drop-down menu "**Scale**" above each graph.

Once the desired datasets have been selected, the list of instances can be hidden by clicking on the "**Hide Instance Selection**" button.

12 Settings


Live Capture running on Ethernet Interface.
Start
Dashboard
Traffic
Scopes
Devices
Analysis
Settings
MEINBERG
Apply
Reset
Packet Capture / Remote Capture / Notifiers & Alarms / Statistics / Web Server / Terminology / User Management / Memory Usage / Logging
Add
OK
Cancel

Scopes 1
Devices 7
Instances 7

All Messages 119.913
Buffered 119.913
Announce Messages 4.936

PTP Track Hound 2.1.2
Licensed to: Darryl Le Count (Professional)
Logged in as: trackhound (rw)

Packet Capture

Interfaces

enp1s0	☐
Addresses	172.16.101.6/16, fe80::cd8f:f2f3:8271:83c1/64
enp2s0	✓
Addresses	192.168.38.1/24, fe80::3c93:64f4:51a0:7b08/64
Alias	
Assign Segment-ID	☐
Filters	
Protocol	any
Version	any
PTPv1 Subdomain	☐
PTPv2 Domain	☐
Enable Management Messages	✓

Statistics

Maximum Entries (per Instance)	300

Tresholds

Metric	Severity	Threshold	Instances
Measured Offset	Warning	40 ns	

Metric	
Severity	
Threshold (ns)	
Instances	-

Figure 12.1: Settings

The **Settings** section allows PTP Track Hound v2 to be fully configured to meet your needs. Please note that some of these options require a **Capture**, **Basic**, or **Professional License**; this requirement is indicated where necessary.

12.1 Packet Capture

The **Packet Capture** window is used to configure the capture of PTP traffic through local network interfaces and also through remote PTP Track Hound v2 instances.

Interfaces

This shows each of the physical network interfaces present on the local device on which this PTP Track Hound v2 instance is running. Each interface is represented by a checkbox and is accompanied by the friendly name and description provided by the underlying operating system via the Npcap driver.

Enabling any one of these interfaces will display further options to enable further configuration:

Alias	The network interface is normally represented in logging and on-screen notifications by the default **Description** field under Windows, and by the device name under Linux and macOS. This description may optionally be overridden by specifying an alternative alias in this field.
Assign Segment ID	A segment ID can be assigned to a local network interface to further break down traffic in a common scope into segments defined by their traffic source. This can be useful, for example, if a single PTP Track Hound v2 instance is to be used to monitor PTP traffic through two individual subnets that are unreachable to one another. Segment IDs can also used to isolate traffic from external PTP Track Hound v2 instances (see "**Remote Connections**" below). For more information on segmentation, refer to Chapter 9, "**Scopes**".

Filters

Filters can be used to limit the traffic accepted by the network interface to specific types.

Protocol	This can be used to limit captured data to the IEEE 802.3, IPv4, or IPv6 network protocol.
Version	This can be used to limit captured data to a specific PTP version, namely PTPv1, PTPv2, or PTPv2.1.
PTPv1 Subdomain	If enabled, this can be used to limit captured PTPv1 data to a specific PTPv1 subdomain.
PTPv2 Domain	If enabled, this can be used to limit captured PTPv2 or PTPv2.1 data to a specific PTPv1 subdomain.

Enable Management Messages

PTP management messages can be used to acquire more detailed information directly from PTP clocks in the network to enable a more in-depth analysis.

Important!

This feature requires a **Basic License**.

Interval	This specifies how frequently management messages will be sent to the PTP multicast address. More frequent management messages can improve the quality of data analysis but also risk overwhelming your network with large amounts of multicast traffic.
Protocol	This specifies whether management messages should only be sent out over a specific protocol (*IPv4, IPv6, IEEE 802.3*). This can be useful for reducing the volume of multicast traffic.
IPv6 Scope	This option only appears when sending management messages in an IPv6 address space. This option specifies the addresses to which the management messages will be sent.
PTP Version	This specifies whether management messages should only be sent out over a specific PTP version (*PTPv1, PTPv2, PTPv2.1*). This can be useful for reducing the volume of multicast traffic.
Specific PTPv1 Subdomain	This specifies whether PTPv1 management messages should only be sent to PTPv1 clocks in a defined PTPv1 subdomain. If disabled, management messages will be sent to all of the commonly used subdomains (*_DFLT, _ALT1, _ALT2, _ALT3*).
Specific PTPv2 Domain	This specifies whether PTPv2 or PTPv2.1 management messages should only be sent to PTPv2/PTPv2.1 clocks in a defined PTPv2/v2.1 domain. If disabled, management messages will be sent to all PTPv2 domains (*0–255*).
Loopback	Enabling this will also send management messages to the local loopback interface, thus ensuring that they are logged in the capture log of the local PTP Track Hound v2 instance.

Enable Capture Time Offsets (CTO)

If enabled, the timestamps of incoming Sync and Follow Up Messages are compared against the corresponding capture timestamp of the message, and the offset between these timestamps is calculated. This data is stored for statistical analysis and provides the basis for CTO threshold events (see below). The clock that provides the capture timestamps (whether the system time of the PTP Track Hound v2 host device or the hardware clock of the network adapter) must be synchronized to enable plausible capture time offsets to be calculated.

Important!

This feature requires a **Basic License**.

Enable NetSync Monitor (NSM)

If enabled, known instances are polled using special NetSync Monitor messages to allow direct measurement of the offset and path delay. This is done by requesting that the measured node send a Delay Response message, followed by a Sync message and (if the node is operating in two-step mode) a Follow Up message to provide the instance's own time.

Important!

This feature requires a **Basic License**, and also requires that the participating nodes support the NetSync Monitor protocol.

Monitor Known Nodes	If enabled, PTP Track Hound v2 will send NetSync Monitor Delay Request messages to all known nodes in accordance with the specified frequency, protocol, and domain settings.
Interval (s)	If **Monitor Known Nodes** is enabled, this is the frequency at which NetSync Monitor Delay Request messages will be sent.
Protocol	If **Monitor Known Nodes** is enabled, NetSync Monitor Delay Request messages can be sent either to all known IEEE802.1, IPv4, and IPv6-addressed nodes (*any*) or only to nodes addressed via IPv4 or IPv6.
Domain	If **Monitor Known Nodes** is enabled, NetSync Monitor Delay Request messages can be sent either to all PTPv2 domain numbers or limited to a specific domain.

Specific Nodes

PTP Track Hound v2 can send NetSync Monitor Delay Request messages not only to nodes that have already been detected but also to nodes at a specific address. This may be necessary due to PTP Track Hound v2 being unable to detect a node automatically for whatever reason, and can also be done to limit the transmission of NetSync Monitor packets to individual nodes.

To define a specific node that will receive NetSync Monitor packets, first click on the "**Add**" button just beneath the **Specific Nodes** table and complete the fields that are enabled:

Address	The MAC, IPv4, or IPv6 address of the PTP instance to which NetSync Monitor Delay request messages are to be sent.
Interval (s)	The frequency at which NetSync Monitor Delay Request messages will be sent to this specific node.
Domain	The PTPv2 domain of the specific node.

Once this information has been added, click on the green "**OK**" button to add the node. You may then add another node as necessary.

Important!

Remember to click on the green "**Apply**" button at the top of the page when you have finished configuring specific instances, otherwise your configuration will not be saved!

<u>Remote Connections</u>

Remote connections are used to receive traffic sent by one or more remote PTP Track Hound v2 instances (with Basic Licenses) on another device. The remote instances must be correspondingly configured; refer to Chapter 12.2, "**Remote Capture**" for more information.

Important!

This feature requires a **Professional License**.

To define a remote connection, first click on the "**Add**" button at the top right of the window, then enter the relevant details:

Host	The host sending the capture data. This can be a fixed IP address or a hostname.
Port	The port on which the remote instance is sending the capture data. This is set to *32676* by default but can be changed to any free port.
Encryption	If you wish to encrypt the transmission of capture data, you may select *AES-256* encryption. In this case, you will need to use a shared key on both the remote instance (configured under "**Remote Capture**") and this instance. The "**Generate Random Key**" button can be used to generate a random 256-bit key (32 characters); this same key should then be copied to the corresponding field under "**Remote Capture**" on the remote instance.
Alias	If a name is entered here, this alias will be used to represent the remote instance in logs and events.
Assign Segment ID	A segment ID can be assigned to capture data from a remote instance to further break down traffic in a shared scope into segments defined by their traffic source. This can be useful, for example, if two difference PTP Track Hound v2 instances are used to monitor PTP traffic in two different subnets that are unreachable to one another, but which would be normally assigned to the same scope due to their using having the same PTP version, (sub)domain and network protocol. For more information on segmentation, refer to Chapter 9, "**Scopes**".

<u>Miscellaneous</u>

Initialization Phase Duration (s)	When the PTP Track Hound v2 capture service is first started, it will wait for the period of time specified here before it actually generates events and scopes based on newly detected or changed devices or instances. This ensures that any 'known' clocks & instances are detected 'quietly' upon startup, and helps ensure that

	events relate solely to the detection of 'new' clocks & instances. The default setting of 30 seconds will allow the system to 'settle' before PTP traffic is captured.
Enable Auto Dump	If enabled, captured data will automatically be dumped to a file and written to the local file server, so that it can be downloaded via the Web Interface. This requires the local file server to be configured; refer to Chapter 12.5, "Web Server (Web Interface Only)" for more information.
Enable Dashcam Mode	If enabled, the captured data from the defined period of time prior to a configured event will automatically be dumped to a file and written to the local file server, so that it can be downloaded via the Web Interface. This requires the local file server to be configured; refer to Chapter 12.5, "Web Server (Web Interface Only)" for more information.
Dashcam Recording Duration	This specifies the length of time encompassed by the 'dashcam file' that is saved when a defined event occurs. It may be a value between 15 to 60 seconds.
Dashcam Fileserver URL	This URL is appended to the dashcam filename in notifications and logs. For example, it can be used to specify a protocol, hostname, and/or FQDN to form a functioning download link for the dashcam file.

12.2 Remote Capture

The **Remote Capture** function of PTP Track Hound v2 allows this instance to transfer its locally captured data to another remote PTP Track Hound instance for aggregation and analysis.

Important!

This feature requires a **Basic License**. The receiving instances must be equipped with a **Professional License** and must be configured accordingly to accept the captured data.

Enabled	This enables the remote capture functionality.
Port	The port on which this instance will send the capture data. This is set to *32676* by default but can be set to any free port. The receiving instance must be configured to accept capture data on the same port; refer to Chapter 12.1, "Packet Capture" for more information.

Allowed Clients

To define a remote connection, first click on the "**Add**" button under **Allowed Clients**, then enter the relevant details:

Address	The instance to which the capture data is to be sent. This can be a fixed IP address or a hostname.
Encryption	If you wish to encrypt the transmission of capture data, you may select *AES-256* encryption. In this case, you will need to use a shared key on both this local instance and the receiving instance (configured under "Packet Capture") . The "**Generate Random Key**" button can be used to generate a random 256-bit key (32 characters); this same key should then be copied to the corresponding field under "Packet Capture" on the receiving instance.

Once this information has been added, click on the green "**OK**" button to add the client. You may then add another client as necessary.

Important!

Remember to click on the green "**Apply**" button at the top of the page when you have finished configuring remote clients, otherwise your configuration will not be saved!

Local Packet Evaluation & Statistics	If enabled, this capture data is analyzed and reflected in the local statistical analysis of this PTP Track Hound v2; if disabled, the capture data will simply be collected, forwarded to the receiving instance, and deleted.

12.3 Notifiers and Alarms

PTP Track Hound v2 allows you to receive notifications via email, an SNMP agent, or a syslog solution if certain defined conditions are met. For example, you can be notified via email if the Grandmaster's quality changes or another clock assumes the role of Grandmaster within a scope, as these events may be indicative of problems with the device that you intend to serve as the Grandmaster.

Important!

This feature requires a **Professional License**.

To define a notifier, first click on the "Add" button, then select the type that you wish to define:
SMTP, SNMP, or syslog.

Important!

Remember to click on the green "**Apply**" button at the top of the page when you have finished configuring notifiers & alarms, otherwise your configuration will not be saved!

SMTP

Smarthost	The SMTP smarthost address.
Port	The port for the SMTP smarthost. The default SMTP port is *25*.
Sender	The email address from which email notifications will be sent.
Recipients	The email addresses to which email notifications will be sent. Each address should be entered individually and then confirmed by clicking on the + symbol next to the field. If you wish to remove an address, click on the cross with the pink background next to the corresponding address.
Authentication	If the SMTP smarthost requires authentication, the credentials can be entered here.
Attach Dashcam File	If enabled, a dashcam file containing the captured PTP data from the configured period of time prior to the event will be attached to the email. This requires Dashcam Mode to be enabled and configured; refer to Chapter 12.1, "Packet Capture" for more information.

SNMP

Version	The SNMP version to be used; this will depend on what your SNMP management solution supports.
Receiver Address	The IP address or hostname of the SNMP management solution.
Port	The port used for communication with the SNMP management solution. The default port for SNMP traps is 162.
Community	The community string used for authentication under SNMPv1 and SNMPv2c.
Engine ID	The engine ID used for identification under SNMPv3.
Security Name	The security name used for the access policy under SNMPv3.
Security Level	The security level used for SNMPv3: *noAuthNoPriv*, *authNoPriv*, or *authPriv*. Depending on whether you choose to enable SNMPv3 authentication and/or privacy, you will need to configure the authentication and encryption mechanisms for your SNMP management solution.
Authentication Protocol	Specifies the authentication protocol for SNMPv3 authentication: *MD5*, *SHA1*, *SHA224*, *SHA256*, *SHA384*, or *SHA512*. This requires a security level of *authNoPriv* or *authPriv*.
Authentication Password	The authentication password of the SNMPv3 management solution.
Privacy Protocol	Specifies the privacy protocol for SNMPv3 encryption: *DES*, *AES128*, *AES192*, or *AES256*. This requires a security level of *authPriv*.

Syslog

Server Address	The IP address or hostname of the syslog server.
Port	The port used for communication with the syslog server. The default port for syslog communication is *514*.
Protocol	Specifies whether notifications should be sent to the syslog server via *UDP* or *TCP*.

Event Triggers

Capture Started	Triggers an event when PTP Track Hound v2 begins to capture data. This event is only triggered when the PTP Track Hound v2 service is launched (provided that live capture is configured to start automatically) or when the live capture is started manually.
Capture Stopped	Triggers an event when the live capture of PTP Track Hound v2 is manually terminated. This event is not triggered if the PTP Track Hound v2 service is unexpectedly terminated for any reason.
Scope Detected	Triggers an event when PTP Track Hound v2 detects clock relationships that constitute a new scope. Refer to Chapter 9, "**Scopes**" for more information.
Device Detected	Triggers an event when PTP Track Hound v2 detects a new device.
Port Detected	Triggers an event when PTP Track Hound v2 detects a new port on a device.
Instance Detected	Triggers an event when PTP Track Hound v2 detects a new instance on a device.
Grandmaster Changeover	Triggers an event when a different instance is selected as Grandmaster in a given scope.
Port State Changed	Triggers an event if the port state of an instance changes.
Local Quality Changed	Triggers an event if the local quality of a detected clock changes.
Grandmaster Quality Changed	Triggers an event if the quality of the Grandmaster clock changes. This can be useful for detecting problems with the Grandmaster's upstream reference source, for example.
Uncontinuous Sequence ID	Triggers an event if the sequence IDs of two supposedly consecutive messages are discontinuous (e.g., there is an increment of more than 1, an identical ID, or a lower ID).
Metric Threshold Exceeded	Triggers an event if a threshold value defined under "**Thresholds**" (see Chapter 12.4, "**Statistics**") has been exceeded.
Metric Normalized	Triggers an event if a threshold value defined under "**Thresholds**" is no longer exceeded after having previously been exceeded.
CTO Threshold Exceeded	Triggers an event if the defined threshold value for the Capture Time Offset is exceeded at any time.
CTO Normalized	Triggers an event if the defined threshold value for the Capture Time Offset is no longer exceeded after having previously been exceeded.
CTO Delta Threshold Exceeded	Triggers an event if the defined threshold value for the Capture Time Offset delta is exceeded at any time.
CTO Delta Normalized	Triggers an event if the defined threshold value for the Capture Time Offset delta is no longer exceeded after having previously being exceeded.
Custom Alarm Triggered	Triggers an event if a condition defined under **Custom Alarms** becomes *true*.
Custom Alarm Cleared	Triggers an event if a condition defined under **Custom Alarms** becomes *false* after it was previously *true*

Custom Alarms

Custom alarms can be defined based on parameters that are readable as JSON output via the PTP Track Hound v2 REST API. To define a custom alarm, first click on the "Add" button, then select the type that you wish to define: SMTP, SNMP, or syslog.

Severity This defines the severity under which events will be logged, for example in logfiles or notifications to a syslog server.

Condition The condition is specified in three parts: the parameter from the REST API relative to the API root `/api`, a logical operator, and the operand to which the parameter will be compared. Chapter 13, "**REST API Reference**" provides a detailed list of all resources available via the REST API.

For example, setting **condition** to `/api/current/instances/0/utcOffset`, **operator** to *is greater than*, and **comparison value** to `37` will trigger a custom alarm whenever the UTC offset of the instance 0 exceeds 37 seconds.

Restrictions If the API resource specified in **Condition** above is a wildcard path that refers to multiple resources, restrictions can be specified here that limit which resources the condition is applied to. Click on the green + symbol to add a new restriction, or on the cross with the pink background to remove an existing one.

For example, the path `/api/current/instances/*/utcOffset` refers to the UTC offset values of every instance known to PTP Track Hound v2. Specifying a restriction of */state* equals "*Master*" will ensure that alarms will only be triggered for instances with a port state of "*Master*" when that path is used for a comparison.

Description The description consists of two components: a parameter name, and a JSON pointer to the object.

For example, if a custom alarm refers to `/api/current/instances/*/localQuality/clockAccuracyValue`, a suitable description would be "*Clock Accuracy*" for description and */address* as the object, which will output "*Clock Accuracy of 192.168.20.20*" to log files, for example.

12.4 Statistics

The **Statistics** tile of the PTP Track Hound v2 Settings page is used to configure how statistics derived from analyses using measurement messages, Capture Time Offsets and NetSync Monitor are handled and to set up alarm thresholds based on these.

Important!

This feature requires a **Basic License**.

Maximum Entries (per Instance)	The maximum number of values that can be stored per instance for management message reports, Capture Time Offset measurements, and NetSync Monitor measurements. This limit applies to each **type** of statistical value, i.e., the default value of *300* allows 300 data points based on management messages, another 300 based on Capture Time Offset measurements, and another 300 still based on NetSync Monitor measurements. With management message reports and NetSync Monitor measurements, the time period encompassed by the data is calculated as: *Maximum entries per instance* $\times$ *polling interval* = *time horizon* This means that, if the maximum number of entries is set to *300* and management messages are sent every 60 seconds, five hours of reported data will be stored. With Capture Time Offset calculations, the time period encompassed by the data is calcuated as: *Maximum entries per instance* $\times$ *sync interval of instance* = *time horizon* This means that, if the maximum number of entries is set to *300* and the instance is configured to send a Sync message once per second, five minutes of data will be stored.

Thresholds

To define a threshold, first click on the "Add" button, then select the type that you wish to define:

Reported Offset	The clock offset as reported by the timeReceiver itself (via PTP management messages).
Reported Offset Delta	The difference between the current and previous clock offset measurements as reported by the timeReceiver itself (via PTP management messages).
Reported Delay	The path delay as reported by the timeReceiver itself (via PTP management messages).
Reported Delay Delta	The difference between the current and previous path delay measurements as reported by the timeReceiver itself (via PTP management messages).
Measured Offset	The clock offset as measured by PTP Track Hound v2 (via NetSync Monitor).

Measured Offset Delta	The difference between the current and previous clock offset measurements as measured by PTP Track Hound v2 (via NetSync Monitor).
Measured Delay	The path delay as measured by PTP Track Hound v2 (via NetSync Monitor).
Measured Delay Delta	The difference between the current and previous path delay measurements as measured by PTP Track Hound v2 (via NetSync Monitor).
CTO	The difference between the timestamp provided in the Sync or Follow Up message from the clock and the Capture Timestamp generated by PTP Track Hound v2 based on the local clock.
CTO Delta	The change in the current and previous Capture Time Offset measurement.

Severity	The severity of the log message output to logging and messaging channels: *Error*, *Warning*, *Info*, *Debug*, or *Trace*.
Threshold (ns)	The threshold in nanoseconds at which the event or alarm is to be triggered.
Instances	If you wish to have this threshold alarm triggered only for certain PTP instances, the IPv4, IPv6, or MAC address of these instances should be entered here and the + should be selected to confirm.

Once the threshold has been defined, click on the green **OK** button at the bottom of the tile.

Individual instances or entire threshold definitions can be removed by clicking on the **X** next to them.

Important!

Remember to click on the green "**Apply**" button at the top of the page when you have finished configuring thresholds, otherwise your configuration will not be saved!

12.5 Web Server (Web Interface Only)

Important!

The Web Server configuration window is only visible when operating PTP Track Hound v2 via the background service and the browser-based Web Interface.

This window contains all settings concerning the web server used to provide the Web Interface and the file server.

HTTP

Enabled	This enables the HTTP server to provide access to the Web Interface and REST API via HTTP.
Port	This is the port via which the Web Interface and REST API will be provided via HTTP. The default port is `8080`, but can be set to any other free port. The port will usually need to be specified through the URL. For example, if a port of `1234` is specified, the Web Interface will be accessible via a browser at `http://127.0.0.1:1234`.

HTTPS

Enabled	This enables the HTTPS server to provide access to the Web Interface and REST API via HTTPS.
Port	This is the port via which the Web Interface and REST API will be provided via HTTPS. The default port is `443`, but can be set to any other free port. If any other port than 443 is used, the port will need to be specified through the URL. For example, if `1234` is specified as the port, the Web Interface will be accessible via a browser at `https://127.0.0.1:1234`.
Certificate	The filename of a valid SSL/TLS certificate. If no absolute path is specified, the file is assumed to be located in the PTP Track Hound v2 installation directory. Certificates can be generated using the included `trackhound-certgen` tool if one is not already available.
Private Key	The filename of the private key file for the SSL/TLS certificate. If no absolute path is specified, the file is assumed to be located in the PTP Track Hound v2 installation directory. This private key file can be generated together with the SSL/TLS certificate using the included `trackhound-certgen` tool.

Important!

Please note that HTTPS cannot be enabled without a valid certificate and private key file.

It is essential that either HTTP or HTTPS remain enabled to provide access to the Web Interface. If both interfaces are disabled, the only way to re-enable them will be to edit the PTP Track Hound v2 configuration file directly.

Therefore, if you wish to disable HTTP, please ensure beforehand that HTTPS access is functioning correctly and that the renewal of the corresponding TLS/SSL certificate is scheduled accordingly.

File Server

The file server is used to host capture files that have been imported from external sources such as Wireshark or dumped by means of the Auto Dump and Dashcam functions. It is also used to host image files used as vendor logos on the **Scopes** and **Devices** pages.

Enabled	Enables the file server.
Directory	The path, relative to the local device, under which files will be saved and served by PTP Track Hound v2.

Capture Files

Capture files that have been imported from an external or dumped by means of the Auto Dump and Dashcam functions are listed here.

Image Files

Image files that have been uploaded for use as vendor logos are listed here. New logos can be uploaded to the file server by clicking on the "**Import**" button and selecting the image from the file selector, or manually copied to the relevant `img` directory.

Individual capture and image files can be deleted by clicking on the cross with the pink background next to the file in question. They may also be manually deleted them from the relevant directory.

12.6 Files (Solo Mode Only)

This window contains settings regarding file handling under Solo Mode.

Capture Files

Capture files that have been imported from an external or dumped by means of the Auto Dump and Dash-cam functions are listed here.

Image Files

Image files that have been uploaded for use as vendor logos are listed here.

Under Windows, these files are located under `%localappdata%\trackhound-solo\solo-files`.

Under Linux and macOS, these files are located under `$HOME\.trackhound-solo\solo-files`. Note that because PTP Track Hound v2 needs to be run with root permissions, `$HOME` in this case will usually be `\root` if it is run under an account lacking root permissions using `sudo`.

New logos can be added to the file storage by clicking on the "**Import**" button and selecting the image from the file selector, or manually copied to the relevant `img` directory.

Individual image and capture files can be deleted by clicking on the cross with the pink background next to the file in question. They may also be manually deleted them from the relevant directory.

12.7 Terminology

By default, PTP Track Hound v2 uses the original IEEE 1588-2009 terminology to describe port states of clocks; as such, ports are described by default as Slave, Pre-Master, or Master.

With discussions ongoing regarding the suitability of this terminology in a modern social & political context, this window provides the option to define alternative port state designations. The initial configuration process will suggest timeReceiver, Pre-timeTransmitter, and timeTransmitter as mappings.

This is a purely aesthetic change and does not affect the actual function of PTP Track Hound v2.

To add a new mapping, click on the "**Add**" button, select the default IEEE 1588-2019 port state designation, and enter the desired terminology in the field "**Mapping**". To confirm, click on the green "**OK**" button.

Important!

Remember to click on the green "**Apply**" button at the top of the page when you have finished configuring terminology, otherwise your configuration will not be saved!

To remove a mapped term, click on the cross with the pink background to the right of the term.

12.8 User Management

This tile is used to create user accounts for the Web Interface and REST API. Each given account can be assigned or denied write access; an account without write access cannot modify the configuration, upload capture files or images to the file server, stop or start the capture service, export capture data, register a new license, or upload JSON payloads via the REST API.

To create a new user, click on the "**Add**" button. Enter the username and the password. If this user is to have write access, check the "**Write Access**" accordingly. Click on the green "**OK**" button to confirm.

To change a user's password, click on the **Key** symbol and enter the new password.

To delete a user, click on the cross with the pink background. Please note that you will not be prompted to confirm the deletion!

Important!

Remember to click on the green "**Apply**" button at the top of the page when you have finished configuring users, otherwise changes to user accounts will not be saved!

12.9 Memory Usage

The **Memory Usage** tile is used to specify the maximum amount of RAM that PTP Track Hound v2 will use to store PTP capture data. When this buffer is nearly full, older data will be purged from memory to make space for the newer capture data.

Naturally, the higher the value specified in "**Maximum Memory Usage**", the more RAM will be required by the system. PTP Track Hound v2 will not allocate this entire amount of RAM immediately; instead, it will allocate it in chunks as needed as specified under "**Chunk Allocation Size**".

As of v2.1.0, the **Maximum Memory Usage** can be set to a valid between 16 MB and 8 GB. By default, is *256 MB* and the **Chunk Allocation Size** is *16 MB*. This means that a maximum of 256 MB RAM will be used for capture data and analysis, and that 16 MB of RAM will be allocated at a time whenever the currently allocated RAM is insufficient.

Lowering the **Chunk Allocation Size** allows for more flexible use of RAM, but can also cause performance to suffer as the system has to assign more resources to a more regular dynamic allocation and freeing up of RAM. The chunk size can be between 256 KB and 16 MB.

12.10 Logging

Standard Streams (i.e., stdout/stderr)

This controls the output to the standard output streams, typically the console from which the PTP Track Hound v2 service is launched. These streams can of course be redirected to another destination such as a file, but PTP Track Hound v2 provides specific functions for file log output and syslog support.

Logfile

This controls the output to a text logfile, which may be located on a local or network drive.

Syslog (Linux/macOS)

This controls the output to the local syslog server on Linux or macOS systems.

Event Log (Windows)

This controls the output to Windows' native Event Log. PTP Track Hound v2-related events are logged under the source *Cygwin*.

Severity

The severity of log output will dictate how detailed the log output is. The higher the severity, the more detailed the log output, but the larger the log files will be.

info	Successful user logins, user logouts, start and termination of capture sessions, erasure of capture data, import of capture files, detection of new PTP devices & instances, creation of new scopes, port state changes, Grandmaster quality changes, local quality changes, Grandmaster changeovers, successful user logins via HTTP Basic Auth, successful user HTTP(S) operations via REST API, normalization of Capture Time Offset values, normalization of Capture Time Offset delta values, custom alarms with severity *info*
warning	Failed user login via Web Interface, due to incorrect username or password, failed user login due to incorrect username or password via REST API (HTTP Basic Auth), failed remote access attempt due to improper license, failed user login due to improper license via REST API (HTTP Basic Auth), Capture Time Offset threshold overruns, Capture Time Offset delta threshold overruns, custom alarms with severity *info* or *warning*
error	Connection errors (e.g., with SMTP servers), sending errors, attempts to GET resources via REST API that do not exist, attempts to PUT resources via REST API via an account without write access, attempts to PUT invalid or inconsistent configuration data via REST API, attempts to set improper configuration parameters via Web Interface, attempts to modify settings via Web Interface via an account without write access, custom alarms with severity *info*, *warning*, or *error*
debug	Sending of SNMP traps, sending of syslog messages, successful HTTP(S) operations via Web Interface, memory allocation/deallocation operations, verbose output relating to start and termination of packet capture, custom alarms with severity *info*, *warning*, *error*, or *debug*
trace	HTTP(S) session validations, periodical network inventory updates, service initialization status (completion time of initialization), all custom alarms

13 REST API Reference

Important!

The use of the PTP Track Hound v2 REST API requires a Professional License.

The integrated REST API provides application developers with the means to reference almost any data from PTP Track Hound v2 in their applications, including captured data, statistics, and system events, and to reconfigure PTP Track Hound v2.

The root endpoint of the REST API is:

```
http://[hostnameOrIPAddress]:[HTTPport]/api
```

or

```
https://[hostnameOrIPAddress]:[HTTPSport]/api
```

For example, from the localhost device with the default port settings, this would be

```
http://127.0.0.1:8080/api
```

or

```
https://127.0.0.1/api
```

.

Authentication and Account Management

Authentication for the REST API is handled via HTTP/HTTPS basic access authentication ("HTTP Basic Auth"). While this reference assumes a certain degree of security-related competence on the part of the developer in the handling of HTTP Basic Auth, it is important to be aware of certain limitations of HTTP Basic Auth in the context of PTP Track Hound v2:

- When using HTTP Basic Auth instead of HTTPS, account credentials are transmitted in plain Base64 form and are not encrypted or hashed in any way.
- When using HTTPS Basic Auth, account credentials are encrypted using TLS 1.3 and are therefore transmitted securely. However, unless the SSL/TLS certificate is signed by a trusted certificate authority, many HTTP agents such as `wget` and `curl` will reject self-signed certificates such as those generated using `trackhound-certgen`. Meinberg strongly recommends using a certificate that is signed by a trusted Certificate Authority for REST API access. However, the certificate check can be disabled with the HTTP agent's appropriate switch. With `curl`, pass the switch `--insecure` (or `-k`) to achieve this. With `wget`, pass the switch `--no-check-certificate`.
- For security purposes, it is recommended to create a specific PTP Track Hound v2 account for REST API access so that it can be disabled or modified as necessary if the account is compromised for any reason. If the account designated for REST API communication will not be used to modify the configuration of the PTP Track Hound v2 instance, it is recommended that write access be disabled for that account.

Resource Notation

Resources within the PTP Track Hound v2 REST API follow a common theme. They are grouped into nine top-level resources: `cache`, `capture`, `config`, `current`, `events`, `info`, `memory`, `network`, and `files`. The resources beneath these top-level resources vary slightly in their behavior, organization, and access methods, and are therefore addressed individually below. Please note that not all resources have directly referenceable endpoints; many can only be acquired as complete objects via higher-level endpoints. Directly referenceable endpoints are denoted with an asterisk ⋆ in the structure diagrams.

Where a given resource provides an indeterminate number of sub-resources depending on the available data, each of these sub-resources is enumerated with a sequentially assigned numerical index value. So, for example, if the PTP Track Hound v2 capture service has identified 30 devices, these devices will be enumerated as 0–29 under `/api/current/devices/`, and the corresponding endpoints for each of these resources are located beneath these enumerated resources.

For example:

```
/api/current/devices/15/clockID
```

will return the PTP clock ID of the device which has been assigned the enumerated index number `15`.

However, because these enumerated indices may be dynamically reassigned with each restart of the capture service, it may be advisable to reference the fixed ID value; this value is set upon first creation (e.g., first discovery of a device) and retained thereafter. This value is used to reference other resources; instances are associated with their parent devices, for example, using the ID value). These ID values are referenced by specifying the prefix `$`. For example:

```
/api/current/devices/$15/clockID
```

will return the PTP clock ID of the device which has been assigned the ID number `15`.

These enumerated resources can be substituted as necessary with an asterisk wildcard to reference all resources at that level. Therefore:

```
/api/current/devices/*/clockID
```

will return the PTP clock IDs of all devices.

Conventions

- Resource names are always in lower camel case, including abbreviations such as protocol names.
- Strings are enclosed in double quotation marks.
- Integer and floating point values are passed as values without being enclosed in double quotation marks.
- Boolean values are passed as the values `true` or `false` without being enclosed in double quotation marks.
- `GET` queries to directly referenceable endpoints that have no lower-level resources will return unformatted values (with strings in double quotation marks).
- `GET` queries to directly referenceable endpoints that have lower-level resources return JSON-formatted values.
- Payloads are always passed to the API in JSON format using the `PUT` method; this requires a user account with write access.

13.1 /api/cache

The `cache` resources can be used to reference all metadata manually entered for devices, segments, and vendors via the Web Interface. This information can only be queried using the `GET` method and not modified via the REST API.

```
-> /api/cache*
--------> /devices*
-----------------> /[sequential index number]*
---------------------------> /id* (integer)
---------------------------> /clockID* (string)
---------------------------> /custom*
---------------------------------> /alias* (string)
---------------------------------> /firmwareRevision* (string)
---------------------------------> /hardwareRevision* (string)
---------------------------------> /softwareRevision* (string)
---------------------------------> /imagePath* (string)
---------------------------------> /location* (string)
---------------------------------> /modelName* (string)
---------------------------------> /vendor* (string)
--------> /segments*
-----------------> /[sequential index number]*
---------------------------> /id* (integer)
---------------------------> /description* (string)
---------------------------> /name* (string)
--------> /vendors*
-----------------> /[sequential index number]* (integer)
---------------------------> /id* (integer)
---------------------------> /identifiers* (string array)
---------------------------> /imagePath* (string)
---------------------------> /name* (string)
--------> /instances*
-----------------> /[sequential index number]* (integer)
---------------------------> /_index (integer)
---------------------------> /id* (integer)
---------------------------> /address* (string)
---------------------------> /color* (string)
---------------------------> /ctoCount* (integer)
---------------------------> /ctoLatest* (integer)
---------------------------> /delayMechanism* (string)
---------------------------> /device* (string)
---------------------------> /grandmaster (integer)
---------------------------> /grandmasterDevice (integer)
---------------------------> /measuredDelayCount* (integer)
---------------------------> /measuredDelayLatest* (integer)
---------------------------> /measuredOffsetCount* (integer)
---------------------------> /measuredOffsetLatest* (integer)
---------------------------> /parent (integer)
---------------------------> /parentDevice (integer)
---------------------------> /port* (integer)
---------------------------> /reportedDelayCount* (integer)
---------------------------> /reportedDelayLatest* (integer)
```

```
-> /api/cache*
--------> /instances*
-----------------> /[sequential index number]* (integer)
---------------------------> /reportedOffsetCount* (integer)
---------------------------> /reportedOffsetLatest* (integer)
---------------------------> /state* (string)
---------------------------> /timestampMechanism* (string)
---------------------------> /_links*
---------------------------------> /device (string)
---------------------------------> /grandmaster (string)
---------------------------------> /scope (string)
---------------------------------> /self (string)
---------------------------> /custom*
---------------------------------> /address (string)
---------------------------------> /color (string)
---------------------------> /scope* (integer)
---------------------------------> /domain (integer)
---------------------------------> /id (integer)
---------------------------------> /protocol (string)
---------------------------------> /version (string)
---------------------------------> /vlanID (string)
---------------------------> /intervals*
---------------------------------> /announce* (string)
---------------------------------> /delayReq* (string)
---------------------------------> /delayResp* (string)
---------------------------------> /followUp* (string)
---------------------------------> /management* (string)
---------------------------------> /monitoringReq* (string)
---------------------------------> /monitoringResp* (string)
---------------------------------> /pDelayReq* (string)
---------------------------------> /pDelayResp* (string)
---------------------------------> /pDelayRespFollowUp* (string)
---------------------------------> /signalling* (string)
---------------------------------> /sync* (string)
---------------------------------> /syncResp* (string)
---------------------------------> /syncRespFollowUp* (string)
---------------------------> /packetCount
---------------------------------> /announce* (integer)
---------------------------------> /delayReq* (integer)
---------------------------------> /delayResp* (integer)
---------------------------------> /followUp* (integer)
---------------------------------> /management* (integer)
---------------------------------> /monitoringReq* (integer)
---------------------------------> /monitoringResp* (integer)
---------------------------------> /pDelayReq* (integer)
---------------------------------> /pDelayResp* (integer)
---------------------------------> /pDelayRespFollowUp* (integer)
---------------------------------> /signalling* (integer)
---------------------------------> /sync* (integer)
---------------------------------> /syncResp* (integer)
---------------------------------> /syncRespFollowUp* (integer)
---------------------------------> /total* (integer)
```

* Directly referenceable endpoint

13.1.1 /api/cache/devices/[_id]

`/clockID`	*string*	The PTP clock ID of the device.
`/id`	*integer*	The ID set by the PTP Track Hound v2 capture service for this device. This is the value visible under "#" in the **Devices** section of the Web Interface.

13.1.1.1 /api/cache/devices/[_id]/custom

`/alias`	*string*	The alias manually specified for the device.
`/firmwareRevision`	*string*	The firmware revision manually specified for the device.
`/hardwareRevision`	*string*	The hardware revision manually specified for the device.
`/softwareRevision`	*string*	The software revision manually specified for the device.
`/imagePath`	*string*	The image path of the image selected for the device. This is relative to the file server path `/api/config/api/fileServer/directory`.
`/location`	*string*	The location manually specified for the device.
`/modelName`	*string*	The model name manually specified for the device.
`/vendor`	*string*	The vendor manually specified for the device.

13.1.2 /api/cache/segments/[_id]

REST API segment resources are only generated if segments are actually defined. Please note that if the segment ID number for a traffic source is changed, the name and description are not carried over; these remain bound to the original segment ID.

`/description`	*string*	The description of the segment manually entered for that segment ID number under **Scopes**.
`/id`	*integer*	The ID number of the segment as defined in **Settings**.
`/name`	*string*	The name of the segment manually entered for that segment ID number under **Scopes**.

13.1.3 /api/cache/vendors/[_id]

REST API vendor resources are generated on the basis of the vendor IDs of detected clocks.

`/id`	*integer*	The internal ID set by the PTP Track Hound v2 capture service for this vendor. This value is not visible in the Web Interface.
`/identifiers`	*string array*	The various names under which the vendor may be identified as by one of their devices. These may include known misspellings, alternative spellings, and legacy designations (for example, from a provider that has changed name).
`/imagePath`	*string*	The image path of the image selected for the vendor. This is relative to the file server path `/api/config/api/fileServer/directory`.
`/name`	*string*	The standard name used for the Vendor in the Web Interface and logging.

13.1.4 /api/cache/instances/[_id]

REST API instance resources are generated on the basis of the detected instances.

`/_index`	*integer*	The sequentially assigned index value for the instance.
`/address`	*string*	The IPv4, IPv6, or MAC address of the instance.
`/color`	*string*	The default color assigned to this instance for the rendering of graphs. Is provided in RGB hexadecimal format (e.g., `#ff0000` for red).
`/ctoAbs`	*integer*	The current offset in nanoseconds between the PTP timestamp and the time of the local clock. This value is *unsigned* and provides no indication of the direction of the offset.
`/ctoCount`	*integer*	The number of Capture Time Offset data points currently recorded for this instance. This is limited by the value set at `/api/config/statistics/maxEntries`.
`/ctoDelta`	*integer*	The most recent Capture Time Offset delta value (i.e., the difference between the last two offset values) in nanoseconds.
`/ctoHwTimestamp`	*boolean*	Specifies if the local clock timestamp has been generated from a hardware clock.
`/ctoLatest`	*integer*	The host system timestamp (Unix timestamp) of the last Capture Time Offset measurement.
`/ctoNsPrecision`	*boolean*	Specifies whether the Capture Time Offsets are measurable in nanoseconds.
`/ctoReal`	*integer*	The current offset between the PTP timestamp and the time of the local clock as a PTP timestamp. This value is *signed* and therefore indicates the direction of the offset.
`/delayMechanism`	*string*	The delay mechanism used by the instance for measuring propagation delays. Valid values: `E2E`, `P2P`
`/device`	*integer*	The device ID (`/api/current/devices/*/id`) to which this instance is bound.

`/grandmaster`	*integer*	Specifies the instance ID (`/api/current/instances/*/id`) of this instance's Grandmaster. If the instance is itself a Grandmaster, it will refer to its own instance (and by definition will be identical to `/id` below).
`/grandmasterDevice`	*integer*	Specifies the device ID (`/api/current/devices/*/id`) of this instance's Grandmaster. If the instance is itself a Grandmaster, it will refer to its own device (and by definition will be identical to `/device` above).
`/id`	*integer*	The internal ID set by the PTP Track Hound v2 capture service for this instance. This value is for internal use and is not the same as the value displayed in the Web Interface and provided at `/api/cache/instances/[_id]/id`.
`/measuredDelayCount`	*integer*	The number of NetSync Monitor path delay measurement data points recorded for this instance. This is limited by the value set at `/api/config/statistics/maxEntries`.
`/measuredDelayLatest`	*integer*	The host system Unix timestamp of the latest NetSync Monitor path delay measurement data point recorded for this instance.
`/measuredOffsetCount`	*integer*	The number of NetSync Monitor offset measurement data points recorded for this instance. This is limited by the value set at `/api/config/statistics/maxEntries`.
`/measuredOffsetLatest`	*integer*	The host system Unix timestamp of the latest NetSync Monitor clock offset measurement data point recorded for this instance.
`/parent`	*integer*	Specifies the ID of the parent instance (`/api/current/instances/*/id`) of this instance. If this instance is a top-level instance, this value will be `null`.
`/parentDevice`	*integer*	Specifies the ID of the parent instance's device (`/api/current/devices/*/id`) for this instance. If this instance is a top-level instance, this value will be `null`.
`/port`	*integer*	The port number of the instance; this is conventionally appended to the clockID.
`/reportedDelayCount`	*integer*	The number of reported path delay data points recorded for this instance on the basis of Management Messages. This is limited by the value set at `/api/config/statistics/maxEntries`.
`/reportedDelayLatest`	*integer*	The host system Unix timestamp of the latest path delay measurement data point recorded for this instance on the basis of Management Messages.
`/reportedOffsetCount`	*integer*	The number of reported offset measurement data points recorded for this instance on the basis of Management Messages. This is limited by the value set at `/api/config/statistics/maxEntries`.
`/reportedOffsetLatest`	*integer*	The host system Unix timestamp of the latest clock offset measurement data point recorded for this instance on the basis of Management Messages.
`/scope`	*integer*	The scope ID to which this instance has been assigned or the instance properties that are used to determine the scope.

`/state`	*string*	The port state of this instance. The strings here reference the terminology defined at `/api/config/terminology/states/*/value`
`/timestampMechanism`	*string*	Specifies the timestamping mechanism of this instance. Valid values: `One-Step`, `Two-Step`

13.1.4.1 /api/cache/instances/[_id]/_links

`/device`	*string*	The API endpoint to the device on which the instance is running.
`/grandmaster`	*string*	The API endpoint to the instance that serves as Grandmaster to this instance. If the instance is itself a Grandmaster, it will refer to its own instance.
`/scope`	*string*	The API endpoint of the scope to which this instance has been assigned.
`/self`	*string*	This instance's own API endpoint as a link.

13.1.4.2 /api/cache/instances/[_id]/custom

`/address`	*string*	Specifies a custom alias for this instance to be used in place of the hostname or address.
`/color`	*string*	Specifies a custom color for this instance for the rendering of graphs. Is provided in RGB hexadecimal format (e.g., `#ff0000` for red).

13.1.4.3 /api/cache/instances/[_id]/scope

The contents of this API resource are only accessible via the root endpoint `/api` and the endpoint `/api/cache`. They contain information on the properties of the instance used to determine the scope within which the instance is placed.

`/domain`	*integer*	Specifies the PTPv2/PTPv2.1 domain number of the instance.
`/id`	*integer*	The unique ID of the device assigned by PTP Track Hound v2.
`/protocol`	*string*	The network protocol employed by this instance. Valid values: `IPv4`, `IPv6`, `IEEE 802.3`
`/version`	*string*	The PTP version employed by this instance. Valid values: `PTPv1`, `PTPv2`, `PTPv2.1`
`/vlanID`	*string*	The VLAN tag employed by this instance. If there is no VLAN tag, this will be `none`.

13.1.4.4 /api/cache/instances/[_id]/intervals

All resources under /api/cache/instances/*/intervals support the same valid values:

`128/s, 64/s, 32/s, 16/s, 8/s, 4/s, 2/s, 1/s, 1/2s, 1/4s, 1/8s, 1/16s, 1/32s, 1/64s, 1/128s, n/a`

`/announce`	*string*	Specifies the *Announce* message interval of the instance.
`/delayReq`	*string*	Specifies the *Delay Request* message interval of the instance.
`/delayResp`	*string*	Specifies the *Delay Response* message interval of the instance.
`/followUp`	*string*	Specifies the *Follow Up* message interval of the instance.
`/management`	*string*	Specifies the *Management* message interval of the instance.
`/monitoringReq`	*string*	Specifies the *Monitoring Request* message interval of the instance. This relates to NetSync Monitor measurements and will read `n/a` if the instance does not support NetSync Monitor or does not report such an interval.
`/monitoringResp`	*string*	Specifies the *Monitoring Response* message interval of the instance. This relates to NetSync Monitor measurements and will read `n/a` if the instance does not support NetSync Monitor or does not report such an interval.
`/pDelayReq`	*string*	Specifies the *Peer Delay Request* message interval of the instance.
`/pDelayResp`	*string*	Specifies the *Peer Delay Response* message interval of the instance.
`/pDelayRespFollowUp`	*string*	Specifies the *Peer Delay Response Follow Up* message interval of the instance.
`/signalling`	*string*	Specifies the *Signaling* message interval of the instance.
`/sync`	*string*	Specifies the *Sync* message interval of the instance.
`/syncResp`	*string*	Specifies the *Sync Response* message interval of the instance. Relates to *flashPTP* nodes.
`/syncRespFollowUp`	*string*	Specifies the *Sync Response Follow Up* message interval of the instance. Relates to *flashPTP* nodes.

13.1.4.5 /api/cache/instances/[_id]/packetCount

`/announce`	*integer*	Specifies the total number of *Announce* messages counted for this instance since last erasure.
`/delayReq`	*integer*	Specifies the total number of *Delay Request* messages counted for this instance since last erasure.
`/delayResp`	*integer*	Specifies the total number of *Delay Response* messages counted for this instance since last erasure.
`/followUp`	*integer*	Specifies the total number of *Follow Up* messages counted for this instance since last erasure.
`/management`	*integer*	Specifies the total number of *Management* messages counted for this instance since last erasure.
`/monitoringReq`	*integer*	Specifies the total number of *Monitoring Request* messages counted for this instance since last erasure. This relates to NetSync Monitor measurements.
`/monitoringResp`	*integer*	Specifies the total number of *Monitoring Response* messages counted for this instance since last erasure. This relates to NetSync Monitor measurements.
`/pDelayReq`	*integer*	Specifies the total number of *Peer Delay Request* messages counted for this instance since last erasure.
`/pDelayResp`	*integer*	Specifies the total number of *Peer Delay Response* messages counted for this instance since last erasure.
`/pDelayRespFollowUp`	*integer*	Specifies the total number of *Peer Delay Response Follow Up* messages counted for this instance since last erasure.
`/signalling`	*integer*	Specifies the total number of *Signaling* messages counted for this instance since last erasure.
`/sync`	*integer*	Specifies the total number of *Sync* messages counted for this instance since last erasure.
`/syncResp`	*integer*	Specifies the total number of *Sync Response* messages counted for this instance since last erasure. Relates to *flashPTP* nodes.
`/syncRespFollowUp`	*integer*	Specifies the total number of *Sync Response Follow Up* messages counted for this instance since last erasure. Relates to *flashPTP* nodes.
`/total`	*integer*	Specifies the total number of PTP messages counted for this instance since last erasure.

13.2 /api/captures/[_id]

The captures resources can be used to reference all active capture processes, whether via network interfaces or remote connections. Please note that these resources can only be fetched as a complete JSON object from /api/captures.

```
-> /api/captures*
------------> /[sequential index number]
---------------------> /_index (integer)
---------------------> /id (integer)
---------------------> /file (string)
---------------------> /alias (string)
---------------------> /name (string)
---------------------> /encKey (string)
---------------------> /encType (string)
---------------------> /host (string)
---------------------> /port (integer)
---------------------> /type (string)
---------------------> /hardwareTimestamps (boolean)
---------------------> /captureTimeOffsets (boolean)
---------------------> /management
-----------------------------> /enabled (boolean)
-----------------------------> /flood (boolean)
-----------------------------> /interval (integer)
-----------------------------> /ipv6MulticastScope (string)
-----------------------------> /loopback (boolean)
-----------------------------> /protocol (string)
-----------------------------> /subdomain (string)
-----------------------------> /domain (integer)
-----------------------------> /version (string)
---------------------> /netSyncMonitor
-----------------------------> /enabled (boolean)
-----------------------------> /knownNodes
-------------------------------------> /domain (integer)
-------------------------------------> /enabled (boolean)
-------------------------------------> /interval (integer)
-------------------------------------> /protocol (string)
-----------------------------> /specificNodes
-------------------------------------> /[sequential index number]
---------------------------------------------> /address (string)
---------------------------------------------> /domain (integer)
---------------------------------------------> /interval (integer)
```

* Directly referenceable endpoint

/_index	*integer*	The sequential index number assigned in the resource level above.
/id	*integer*	The ID set by the PTP Track Hound v2 capture service for this capture source. This value is not visible in the Web Interface.
/file	*string*	The filename of an externally imported capture file.
/alias	*string*	The alternative alias optionally manually defined via the Web Interface for this capture source.
/name	*string*	The name used for the capture instance in the Web Interface and logging if no alias is defined.

`/encType`	*string*	The type of encryption used for remote capture instances. Valid values: `none`, `aes-256`.
`/encKey`	*string*	The encryption key used for remote capture instances.
`/host`	*string*	The configured host for remote capture instances.
`/port`	*integer*	The configured port for remote capture instances.
`/type`	*string*	The type of capture instance. Valid values: `interface` (for local network interface), `remote` (for capture on a remote instance), `file` (for imported capture file).
`/hardwareTimestamps`	*boolean*	Specifies whether the capture timestamps collected for each message on this local capture source are generated at network controller level by a hardware timestamper.
`/captureTimeOffsets`	*boolean*	Specifies whether Capture Time Offset timestamping is enabled.

13.2.1 /api/captures/[_id]/management

The `management` endpoints are only available for capture instances on network interfaces.

`/enabled`	*boolean*	Specifies whether PTP management messages are enabled for this network interface. Not applicable to remote capture instances.
`/flood`	*boolean*	If `true`, all messages are sent to all domains, via all network protocols, and across all PTP versions simultaneously. If `false`, the service will wait for a time between each network protocol and PTP version in order to reduce bandwidth demand.
`/interval`	*integer*	The frequency in seconds at which PTP management messages are sent out.
`/ipv6MulticastScope`	*string*	The multicast scope for the sending of PTP management messages if IPv6 addressing is used. Valid values: `0x01`, `0x02`, `0x03`, `0x04`, `0x05`, `0x08`, `0x0e`
`/loopback`	*boolean*	Specifies whether PTP management messages sent by this PTP Track Hound v2 instance are looped back into the instance's own capture data.
`/protocol`	*string*	The protocol via which PTP management messages are exclusively sent via this capture instance. Valid values: `IPv4`, `IPv6`, `IEEE802.3`, `any`.
`/subdomain`	*string*	The PTPv1 subdomain over which PTP management messages are exclusively sent via this capture instance. Any PTPv1 subdomain may be specified, including custom subdomains, but the value `any` here means that management messages will only be sent to the four standard subdomains `_DFLT`, `_ALT1`, `_ALT2`, `_ALT3`, `_ALT4` as defined by IEEE1588-2002.
`/domain`	*integer*	The PTPv2/PTPv2.1 domain over which PTP management messages are exclusively sent via this capture instance. A value of `-1` refers to all domains. Valid values: `0-255`, `-1`.

/version	*string*	The PTP version for which PTP management are exclusively sent via this capture instance. Valid values: PTPv1, PTPv2, PTPv2.1, any

13.2.2 /api/captures/[_id]/netSyncMonitor

The netSyncMonitor endpoints are only available for capture instances on network interfaces.

/enabled	*boolean*	Specifies whether NetSync Monitor is enabled for this network interface. Not applicable to remote capture instances.

13.2.2.1 /api/captures/[_id]/netSyncMonitor/knownNodes

/enabled	*boolean*	Specifies whether NetSync Monitor is enabled for nodes that have already been detected by PTP Track Hound v2.
/domain	*integer*	The PTPv2/PTPv2.1 domain over which PTP NetSync Monitor Delay Request messages to sent to all known nodes. A value of −1 refers to all domains. Valid values: 0–255, −1.
/interval	*integer*	The frequency in seconds that NetSync Monitor Delay Request messages are sent to all known nodes.
/protocol	*string*	The protocol via which NetSync Monitor Delay Request messages are sent to known nodes via this capture instance. Valid values: IPv4, IPv6,IEEE802.3, any.

13.2.2.2 /api/captures/[_id]/netSyncMonitor/specificNodes/[id]

This API route provides information about specific custom nodes (nodes that have not been detected by PTP Track Hound v2 and are polled manually) that have been defined for use with NetSync Monitor.

/address	*boolean*	The IPv4, IPv6, or MAC address of the instance to which NetSync Monitor Delay Request messages will be sent.
/domain	*integer*	The PTPv2/PTPv2.1 domain over which PTP NetSync Monitor Delay Request messages are sent to this specific node. Valid values: 0–255.
/interval	*integer*	The frequency in seconds that NetSync Monitor Delay Request messages are sent to this specific node.

13.3 /api/config

The `config` resources can be used to query and modify the current PTP Track Hound v2 configuration.

Please note that these resources can only be fetched or sent as a complete JSON object payload from or to `/api/config`. Therefore, if you wish to modify individual configuration parameters, the entire `/api/config` JSON object must first be acquired, the corresponding parameter in the object modified, and the entire JSON object then sent back to the REST API.

The PTP Track Hound v2 service will verify that the JSON payload is properly formatted before accepting it. Attempting to PUT an improperly formatted JSON payload to `/api/config` will result in the payload being rejected and `400 Bad Request` being returned.

Important!

JSON payloads in `/api/config` can contain confidential information such as SMTP smarthost passwords, SNMP authentication passwords, and SNMP privacy passwords unencrypted as plain text. The use of HTTPS is strongly recommended for the handling of these resources!

Important!

Be careful when sending a configuration data payload to the PTP Track Hound v2 instance via the REST API! The creation of a backup of the original configuration file is recommended!

While the PTP Track Hound v2 service will verify the integrity and formatting of the configuration data and reject overtly invalid values, a payload that contains improper configuration data may cause the PTP Track Hound v2 capture service to crash!

You should only send a complete `/api/config` JSON object! The PTP Track Hound v2 service will reset any unspecified parameters to their default settings, so that the omission of both `/api/config/api/http` and `/api/config/api/https` will result in both protocols being disabled and the service remaining inaccessible as a result until re-enabled locally.

The omission of user account details (including hashed passwords and salts) will result in all user accounts being deleted; the default `trackhound` account will be restored in this case.

If the configuration data explicitly disables REST API's HTTP/HTTPS protocol, modifies its own user account, or contains improper license information, the HTTP agent may be unable to communicate with the PTP Track Hound v2 instance via the REST API until the configuration is corrected by some other means!

```
-> /api/config*
---------> /api
-----------------> /fileServer
----------------------> /enabled (boolean)
----------------------> /directory (string)
-----------------> /http
----------------------> /enabled (boolean)
----------------------> /port (integer)
-----------------> /https
----------------------> /enabled (boolean)
----------------------> /cert (string)
----------------------> /key (string)
----------------------> /port (integer)
---------> /capture
-----------------> /autoDump (boolean)
-----------------> /initDuration (integer)
-----------------> /file (string)
-----------------> /dashcam
----------------------> /enabled (boolean)
----------------------> /duration (integer)
----------------------> /url (string)
-----------------> /interfaces
----------------------> /[sequential index number]
-----------------------------> /alias (string)
-----------------------------> /captureTimeOffsets (boolean)
-----------------------------> /name (string)
-----------------------------> /segmentID (integer)
-----------------------------> /subdomainFilter (string)
-----------------------------> /domainFilter (integer)
-----------------------------> /protocolFilter (string)
-----------------------------> /versionFilter (string)
-----------------------------> /hardwareTimestamps (boolean)
-----------------------------> /management
-----------------------------------> /enabled (boolean)
-----------------------------------> /subdomain (string)
-----------------------------------> /domain (integer)
-----------------------------------> /interval (integer)
-----------------------------------> /ipv6MulticastScope (string)
-----------------------------------> /loopback (boolean)
-----------------------------------> /protocol (string)
-----------------------------------> /version (string)
-----------------------------> /netSyncMonitor
-----------------------------------> /enabled (boolean)
-----------------------------------> /knownNodes
---------------------------------------> /domain (integer)
---------------------------------------> /enabled (boolean)
---------------------------------------> /interval (integer)
---------------------------------------> /protocol (string)
-----------------------------> /specificNodes
---------------------------------------> /[sequential index number]
---------------------------------------------> /address (string)
---------------------------------------------> /domain (integer)
---------------------------------------------> /interval (integer)
```

* Directly referenceable endpoint

```
-> /api/config*
---------> /capture
-----------------> /remoteConnections
----------------------> /[sequential index number]
-----------------------------> /alias (string)
-----------------------------> /host (string)
-----------------------------> /port (integer)
-----------------------------> /segmentID (string)
-----------------------------> /encType (string)
-----------------------------> /encKey (string)
---------> /evaluation
-----------------> /enabled (boolean)
---------> /event
-----------------> /alarms
----------------------> /[sequential index number]
-----------------------------> /condition
-----------------------------------> /left (string)
-----------------------------------> /operator (string)
-----------------------------------> /right (string)
-----------------------------> /description
-----------------------------------> /object (string)
-----------------------------------> /parameter (string)
-----------------------------> /severity (string)
-----------------------------> /where
-----------------------------------> /[sequential index number]
---------------------------------------> /left (string)
---------------------------------------> /operator (string)
---------------------------------------> /right (string)
-----------------> /notifiers
----------------------> /[sequential index number] (integer)
-----------------------------> /events (string array)
-----------------------------> /type (string)
-----------------------------> /config
-----------------------------------> /address (string)
-----------------------------------> /port (integer)
-----------------------------------> /protocol (string)
-----------------------------------> /attachDashcam (boolean)
-----------------------------------> /recipients (string array)
-----------------------------------> /sender (string)
-----------------------------------> /smarthost (string)
-----------------------------------> /authPassword (string)
-----------------------------------> /authProtocol (string)
-----------------------------------> /engineID (string)
-----------------------------------> /privPassword (string)
-----------------------------------> /privProtocol (string)
-----------------------------------> /securityLevel (string)
-----------------------------------> /securityName (string)
-----------------------------------> /community (string)
-----------------------------------> /receiver (string)
-----------------------------------> /version (string)
-----------------------------------> /type (string)
-----------------------------------> /authentication
---------------------------------------> /enabled (boolean)
---------------------------------------> /username (string)
---------------------------------------> /password (string)
```

* Directly referenceable endpoint

```
-> /api/config*
---------> /license
-----------------> /id (string)
-----------------> /key (string)
-----------------> /user (string)
---------> /logging
-----------------> /file
----------------------> /enabled (boolean)
----------------------> /filename (string)
----------------------> /severity (string)
-----------------> /standardStreams
----------------------> /enabled (boolean)
----------------------> /severity (string)
-----------------> /syslog
----------------------> /enabled (boolean)
----------------------> /severity (string)
---------> /memory
-----------------> /chunkSize (integer)
-----------------> /max (integer)
---------> /remote
-----------------> /enabled (boolean)
-----------------> /port (integer)
-----------------> /clients
----------------------> /[sequential index number]
-----------------------------> /address (string)
-----------------------------> /encType (string)
-----------------------------> /encKey (string)
---------> /statistics
-----------------> /maxEntries (integer)
-----------------> /thresholds
----------------------> /[sequential index number]
-----------------------------> /instances (string array)
-----------------------------> /metric (string)
-----------------------------> /severity (string)
-----------------------------> /threshold (integer)
---------> /terminology
-----------------> /states
----------------------> /[sequential index number]
-----------------------------> /value (string)
-----------------------------> /display (string)
---------> /users
-----------------> /[sequential index number]
----------------------> /username (string)
----------------------> /password (string)
----------------------> /salt (string)
----------------------> /writeAccess (boolean)
```

* Directly referenceable endpoint

13.3.1 /api/config/api

13.3.1.1 /api/config/api/fileServer

`/enabled`	*boolean*	Specifies if the file server is enabled.
`/directory`	*string*	The local or network path of the file server. This is referenced by several other resources, including `/api/cache/devices/[index number]/clockID/custom/imagePath`

13.3.1.2 /api/config/api/http

`/enabled`	*boolean*	Specifies if HTTP access is enabled.
`/port`	*integer*	Specifies the port for HTTP access.

13.3.1.3 /api/config/api/https

`/enabled`	*boolean*	Specifies if HTTPS access is enabled.
`/cert`	*string*	The filename of the SSL/TLS certificate. Relative paths are relative to the PTP Track Hound v2 service executable; therefore, if no path is provided, the PTP Track Hound v2 installation directory will be assumed.
`/key`	*string*	The filename of the key file for the SSL/TLS certificate. If no absolute path is specified, this path is relative to the PTP Track Hound v2 service executable.
`/port`	*integer*	Specifies the port for HTTPS access.

13.3.2 /api/config/capture

`/autoDump`	*boolean*	Specifies whether auto-dump is enabled.
`/initDuration`	*integer*	Specifies the initialization wait period in seconds before events are generated.
`/file`	*string*	If a capture file has been selected for analysis, this is the full path to the currently selected capture file.

13.3.2.1 /api/config/capture/dashcam

`/enabled`	*boolean*	Specifies whether Dashcam Mode is enabled.
`/duration`	*integer*	The length of time in seconds that an attached capture file should encompass in Dashcam Mode.
`/url`	*string*	The URL prefix to be appended to the dashcam file path to form a downloadable link.

13.3.2.2 /api/config/capture/interfaces/[_id]

/alias	*string*	The alias manually specified for each network interface.
/name	*string*	Specifies the interface name as set by Npcap (Windows) or the operating system (Linux/Mac).
/segmentID	*integer*	Specifies the segment ID which traffic captured via this interface will be assigned to.
/subdomainFilter	*string*	Specifies the PTPv1 subdomain name if capture traffic via this interface is to be limited to that subdomain. If set to `any`, management messages are sent to all commonly used subdomains (`_DFLT`, `_ALT1`, `_ALT2`, `_ALT3`)
/domainFilter	*integer*	Specifies the PTPv2/PTPv2.1 domain number if capture traffic via this interface is to be limited to that domain.
/protocolFilter	*string*	Specifies the network protocol if capture traffic via this interface is to be limited to that protocol.
/versionFilter	*string*	Specifies the PTP version if capture traffic is to be limited to that version.
/hardwareTimestamps	*boolean*	Specifies whether this interface is configured to use the hardware timestamper within the network controller.
/captureTimeOffsets	*boolean*	Specifies whether Capture Time Offset functionality should be enabled for this interface (`true`) or disabled (`false`).

/management

/enabled	*boolean*	Specifies if management messages are to be sent out via this interface.
/subdomain	*string*	Specifies if management messages should only be sent to a specific PTPv1 subdomain. If set to `any`, management messages are sent to all subdomains.
/domain	*integer*	Specifies if management messages should only be sent to a specific PTPv2/PTPv2.1 domain. If set to `-1`, management messages are sent to all domains.
/interval	*integer*	Specifies how frequently in seconds management messages are to be sent out. Valid values: `15-900` in multiples of 15.
/ipv6MulticastScope	*string*	Specifies the IPv6 scope to which management messages will be multicast in an IPv6 network. Valid values: `0x01`, `0x02`, `0x03`, `0x04`, `0x05`, `0x08`, `0x0e`
/loopback	*boolean*	Specifies whether PTP management messages sent via this interface are to be looped back into the instance's own capture data.
/protocol	*string*	Specifies whether PTP management messages should be sent via this interface over a single protocol only. Valid values: `IPv4`, `IPv6`, `IEEE802.3`, `any`.

`/version`	*string*	Specifies whether PTP management messages should be sent via this interface over a single PTP version only. Valid values: `PTPv1`, `PTPv2`, `PTPv2.1`, `any`

/netsyncMonitor

`/enabled`	*boolean*	Specifies if NetSync Monitor is to be enabled via this interface.

`/knownNodes`

`/enabled`	*boolean*	Specifies if NetSync Monitor Delay Request messages should be sent to known nodes.
`/domain`	*integer*	Specifies if NetSync Monitor Delay Request messages should only be sent to known nodes in a specific PTPv2 domain (*0–255*). If set to `-1`, NetSync Monitor Delay Request messages will be sent to known PTPv2 and PTPv2.1 nodes in all domains.
`/interval`	*integer*	Specifies the rate in seconds at which NetSync Monitor Delay Request messages will be sent to known nodes.
`/protocol`	*string*	Specifies whether NetSync Monitor Delay Request messages should be sent to known nodes via this interface over a single protocol only. Valid values: `IPv4`, `IPv6`, `IEEE802.3`, `any`

`/specificNodes/[_id]`

`/address`	*boolean*	The IPv4, IPv6, or MAC address of the specific node to which NetSync Monitor Delay Request messages should be sent.
`/domain`	*integer*	The PTPv2 domain (*0–255*) of the specific node to which the NetSync Monitor Delay Request messages should be sent.
`/interval`	*integer*	Specifies the rate in seconds at which NetSync Monitor Delay Request messages will be sent to this specific node.

13.3.2.3 /api/config/capture/remoteConnections/[_id]

`/alias`	*string*	The alias manually specified for each remote PTP Track Hound v2 server instance.
`/host`	*string*	The host IP address for the remote PTP Track Hound v2 instance serving the capture data.
`/port`	*integer*	The network port for the remote connection with the PTP Track Hound v2 instance serving the capture data.
`/segmentID`	*string*	Specifies the segment ID which capture data received via this remote connection will be assigned to. Valid values: `0-65534`
`/encType`	*string*	The encryption method for the connection with the remote PTP Track Hound v2 instance serving the capture data. Valid values: `none`, `aes-256`

`/encKey`	*string*	If AES-256 encryption is enabled for this connection, this is the shared secret.

13.3.2.4 /api/config/capture/captureTimeOffset (Disabled)

Information:

The API route `/api/config/capture/captureTimeOffset` introduced in PTP Track Hound v2.0.5 no longer exists as of PTP Track Hound v2.1.0.

The activation state of the Capture Time Offset functionality for each interface can now be read via the route:

`/api/captures/[interface index]/captureTimeOffsets` (endpoint `/api/captures`)

The activation state can be set via the route:

`/api/config/capture/interfaces/[interface index]/captureTimeOffsets` (endpoint `/api/config`).

Thresholds for Capture Time Offsets can be defined and read via the route:

`/api/config/statistics/thresholds/[threshold index]` (endpoint `/api/config`).

Refer to Chapters 13.2, 13.3.2.2, and 13.3.9.1 for more information.

13.3.3 /api/config/evaluation

`/enabled`	*boolean*	If `false`, the local PTP Track Hound v2 instance will not evaluate or analyze the PTP messages captured by itself; it will simply forward the data to the configured remote instances. If `true`, the captured PTP messages will be evaluated and analyzed locally in addition to being forwarded.

13.3.4 /api/config/event

13.3.4.1 /api/config/event/alarms/[_id]

/condition

`/left`	*string*	The API endpoint to be compared.
`/operator`	*string*	The comparison operator. Valid values: `eq`, `ne`, `gt`, `lt`, `ge`, `le`
`/right`	*string*	The value to be compared against.

/description

`/object`	*string*	The API endpoint that provides an identifier for the parameter to be monitored, which will be reported in logs and events alongside `/parameter`. The route in this case is specified relative to the overall object to be monitored. For example, if the API endpoint `/api/current/instances$0/localQuality/clockClass` is to be monitored, PTP Track Hound v2 will automatically identify the object to be monitored as `/api/current/instances/$0`. Any resource at this level or below it can be specified, e.g., `/address` will acquire the identifier at `/api/current/instances/$0/address`, while `/localQuality/clockID` will acquire the identifier from `/api/current/instances/$0/localQuality/clockID`.
`/parameter`	*string*	A description of the parameter to be monitored, which will be reported in logs and events alongside `/object`.

/where/[_id]

`/left`	*string*	The API endpoint, containing a wildcard `*`, specifying a range of resources.
`/operator`	*string*	The comparison operator. Valid values: `eq`, `ne`, `gt`, `lt`, `ge`, `le`
`/right`	*string*	The comparison operand used to limit the resources monitored by the custom alarm.

13.3.4.2 /api/config/event/notifiers/[_id]

`/events`	*string array*	An array containing the events that will cause the notifier to be triggered. Valid values: `captureStarted`, `captureStopped`, `scopeDetected`, `deviceDetected`, `portDetected`, `instanceDetected`, `grandmasterChangeover`, `portStateChanged`, `localQualityChanged`, `grandmasterQualityChanged`, `customAlarmTriggered`, `customAlarmCleared`
`/type`	*string*	The type of notifier receiver. Valid values: `syslog`, `smtp`, `snmp`

/config

/address	*string*	The address of the syslog server. Only applies to syslog servers.
/port	*integer*	The port of the SNMP manager, syslog server, or SMTP server.
/protocol	*string*	Specifies whether syslog messages will be sent over UDP or TCP. Valid values: `udp`, `tcp`
/attachDashcam	*boolean*	Specifies whether to attach a Dashcam file as an email attachment. Only applies to SMTP servers.
/recipients	*string array*	The recipients of email notifications. Only applies to SMTP servers.
/sender	*string*	The specified sender email address of email notifications. Only applies to SMTP servers.
/smarthost	*string*	The SMTP smarthost address. Only applies to SMTP servers.
/authPassword	*string*	The SNMPv3 authentication password. Only applies to SNMPv3 managers with `authPriv` or `authNoPriv` enabled.
/authProtocol	*string*	The hashing algorithm used for SNMPv3 authentication. Only applies to SNMPv3 managers with `authPriv` or `authNoPriv` enabled. Valid values: `MD5`, `SHA1`, `SHA224`, `SHA256`, `SHA384`, `SHA512`
/engineID	*string*	The SNMPv3 engine ID. Only applies to SNMPv3 managers.
/privPassword	*string*	The SNMPv3 privacy password. Only applies to SNMPv3 managers with `authPriv` enabled.
/privProtocol	*string*	The encryption algorithm used for SNMPv3 privacy. Only applies to SNMPv3 managers with *authPriv* enabled. Valid values: `DES`, `AES128`, `AES192`, `AES256`
/securityLevel	*string*	The security level mandated by the SNMPv3 manager. Only applies to SNMPv3 managers. Valid values: `noAuthNoPriv`, `authNoPriv`, `authPriv`.
/securityName	*string*	The security name of the SNMPv3 manager. Only applies to SNMPv3 managers.
/community	*string*	The SNMP community name. Only applies to SNMPv1 or SNMPv2 managers.
/receiver	*string*	The SNMP manager address. Only applies to SNMP managers.
/version	*string*	The SNMP version used by the SNMP manager. Valid values: `SNMPv1`, `SNMPv2c`, `SNMPv3`.
/authentication/enabled	*boolean*	Specifies whether the SMTP smarthost requires authentication. Only applies to SMTP servers.

`/authentication/username`	*string*	The SMTP smarthost authentication username. Only applies to SMTP servers.
`/authentication/password`	*string*	The SMTP smarthost authentication password. Only applies to SMTP servers.

13.3.5 /api/config/license

`/id`	*string*	The registered license ID.
`/key`	*string*	The registered license key.
`/user`	*string*	The registered license name.

13.3.6 /api/config/logging

13.3.6.1 /api/config/logging/file

`/enabled`	*boolean*	Specifies whether logging to file output is enabled.
`/filename`	*string*	The filename of the log output. Relative paths are relative to the PTP Track Hound v2 service executable; therefore, if no path is provided, the PTP Track Hound v2 installation directory will be assumed.
`/severity`	*string*	The maximum severity of log output to file. Valid values: `error`, `warning`, `info`, `debug`, `trace`.

13.3.6.2 /api/config/logging/standardStreams

`/enabled`	*boolean*	Specifies whether logging to `stdout/stderr` is enabled.
`/severity`	*string*	The maximum severity of log output to `stdout/stderr`. Valid values: `error`, `warning`, `info`, `debug`, `trace`.

13.3.6.3 /api/config/logging/syslog

`/enabled`	*boolean*	Specifies whether logging to local syslog/Windows Event Log is enabled.
`/severity`	*string*	The maximum severity of log output to syslog/Windows Event log. Valid values: `error`, `warning`, `info`, `debug`, `trace`.

13.3.7 /api/config/memory

`/chunkSize`	*integer*	The size of each allocable memory chunks in bytes.
`/max`	*integer*	The maximum amount of memory in bytes usable by PTP Track Hound v2 for captured data, analysis, and modeling.

13.3.8 /api/config/remote

`/enabled`	*boolean*	Specifies whether the outbound transmission of capture data is enabled.
`/port`	*integer*	The port over which outbound capture data is sent.

13.3.8.1 /api/config/remote/clients/[_id]

The `clients` are the configured remote PTP Track Hound v2 instances authorized to receive capture data from the local PTP Track Hound v2 instance.

`/address`	*string*	The client IP address for the remote PTP Track Hound v2 instance receiving the local instance's capture data.
`/encType`	*string*	The encryption method for the connection with the remote PTP Track Hound v2 instance receiving the capture data. Valid values: `none`, `aes-256`
`/encKey`	*string*	If AES-256 encryption is enabled for this connection, this is the shared secret.

13.3.9 /api/config/statistics

`/maxEntries`	*integer*	Specifies the number of entries that can be stored for each analysis method (PTP measurement messages, Capture Time Offsets, NetSync Monitor).

13.3.9.1 /api/config/statistics/thresholds/[_id]

`/instances`	*string array*	Specifies the instances (as IPv4, IPv6 or MAC addresses) that this threshold should apply to.
`/metric`	*string*	The metric that the threshold value is measured against. Valid values: `reportedOffset`, `reportedOffsetDelta`, `reportedDelay`, `reportedDelayDelta`, `measuredOffset`, `measuredOffsetDelta`, `measuredDelay`, `measuredDelayDelta`, `cto`, `ctoDelta`
`/severity`	*string*	The severity with which a threshold exceeded event is written to log output. Valid values: `error`, `warning`, `info`, `debug`, `trace`
`/threshold`	*integer*	The value which, when exceeded by the metric, will trigger an alarm.

13.3.10 /api/config/terminology

13.3.10.1 /api/config/terminology/states/[_id]

`/value`	*string*	The standard IEEE port state designation. Valid values: `Initializing`, `Faulty`, `Disabled`, `Listening`, `Pre-Master`, `Master`, `Passive`, `Uncalibrated`, `Slave`
`/display`	*string*	The arbitrary term to replace the standard IEEE port state designation.

13.3.11 /api/config/users/[_id]

`/username`	*string*	The username of the user.
`/password`	*string*	The hashed password of the user.
`/salt`	*string*	The string used to salt the hashed password of the user in transit.
`/writeAccess`	*boolean*	Specifies whether the user has write access to the PTP Track Hound v2 instance. A user with no write access cannot use the `PUT` method to send JSON payloads to the REST API, but can still `GET` payloads.

13.4 /api/current

The `current` resources provide information about the currently detected devices and instances as well as the scopes derived from the analysis of these devices and instances and the manually created segments.

These resources can be referenced at any level, so that entire JSON objects or individual unformatted values can be fetched as needed.

```
-> /api/current*
---------> /devices*
-----------------> /[sequential index number]*
----------------------> /_index (integer)
----------------------> /alias (string)*
----------------------> /alternateVendor (string)*
----------------------> /clockID (string)*
----------------------> /firmwareRevision (string)*
----------------------> /hardwareRevision (string)*
----------------------> /softwareRevision (string)*
----------------------> /id (integer)*
----------------------> /instances (integer array)*
----------------------> /location (string)*
----------------------> /modelName (string)*
----------------------> /ports (integer array)*
----------------------> /type (string)*
----------------------> /vendor (string)*
----------------------> /_links*
-----------------------------> /image (string)
-----------------------------> /self (string)
-----------------------------> /vendorImage (string)
----------------------> /custom*
-----------------------------> /alias (string)*
-----------------------------> /firmwareRevision (string)*
-----------------------------> /hardwareRevision (string)*
-----------------------------> /softwareRevision (string)*
-----------------------------> /imagePath (string)*
-----------------------------> /location (string)*
-----------------------------> /modelName (string)*
-----------------------------> /vendor (string)*
```

* Directly referenceable endpoint

```
-> /api/current*
---------> /instances*
-----------------> /[sequential index number]*
----------------------> /_index (integer)
----------------------> /address (string)*
----------------------> /announceReceiptTimeout (integer)*
----------------------> /color (string)*
----------------------> /ctoAbs (integer)*
----------------------> /ctoCount (integer)*
----------------------> /ctoLatest (integer)*
----------------------> /ctoHwTimestamp (boolean)*
----------------------> /ctoNsPrecision (boolean)*
----------------------> /ctoReal (integer)*
----------------------> /delayMechanism (string)*
----------------------> /device (integer)*
----------------------> /frequencyTraceable (boolean)*
----------------------> /grandmaster (integer)*
----------------------> /grandmasterDevice (integer)*
----------------------> /id (integer)*
----------------------> /illegalMaster (boolean)*
----------------------> /leap59 (boolean)*
----------------------> /leap61 (boolean)*
----------------------> /measuredDelayAbs (integer)*
----------------------> /measuredDelayCount (integer)*
----------------------> /measuredDelayDelta (integer)*
----------------------> /measuredDelayHwTimestamp (boolean)*
----------------------> /measuredDelayLatest (integer)*
----------------------> /measuredDelayReal (integer)*
----------------------> /measuredOffsetAbs (integer)*
----------------------> /measuredOffsetCount (integer)*
----------------------> /measuredOffsetDelta (integer)*
----------------------> /measuredOffsetHwTimestamp (boolean)*
----------------------> /measuredOffsetLatest (integer)*
----------------------> /measuredOffsetReal (integer)*
----------------------> /parent (integer)*
----------------------> /parentDevice (integer)*
----------------------> /port (integer)*
----------------------> /reportedDelayAbs (integer)*
----------------------> /reportedDelayCount (integer)*
----------------------> /reportedDelayDelta (integer)*
----------------------> /reportedDelayLatest (integer)*
----------------------> /reportedDelayReal (integer)*
----------------------> /reportedOffsetAbs (integer)*
----------------------> /reportedOffsetCount (integer)*
----------------------> /reportedOffsetDelta (integer)*
----------------------> /reportedOffsetLatest (integer)*
----------------------> /reportedOffsetReal (integer)*
----------------------> /scope (integer)*
----------------------> /sequenceIDAnnounce (integer)*
----------------------> /sequenceIDRequest (integer)*
----------------------> /sequenceIDSync (integer)*
----------------------> /state (string)*
----------------------> /stepsRemoved (integer)*
```

* Directly referenceable endpoint

```
-> /api/current*
---------> /instances*
-----------------> /[sequential index number]*
----------------------> /timeSource (string)*
----------------------> /timeTraceable (boolean)*
----------------------> /timescale (boolean)*
----------------------> /timestampMechanism (string)*
----------------------> /utcOffset (integer)*
----------------------> /_links*
-----------------------------> /device (string)
-----------------------------> /grandmaster (string)
-----------------------------> /grandmasterDevice (string)
-----------------------------> /parent (string)
-----------------------------> /parentDevice (string)
-----------------------------> /scope (string)
-----------------------------> /self (string)
----------------------> /custom*
-----------------------------> /address (string)*
-----------------------------> /color (string)*
----------------------> /intervals*
-----------------------------> /announce (string)*
-----------------------------> /delayReq (string)*
-----------------------------> /delayResp (string)*
-----------------------------> /followUp (string)*
-----------------------------> /management (string)*
-----------------------------> /monitoringReq (string)*
-----------------------------> /monitoringResp (string)*
-----------------------------> /pDelayReq (string)*
-----------------------------> /pDelayResp (string)*
-----------------------------> /pDelayRespFollowUp (string)*
-----------------------------> /signalling (string)*
-----------------------------> /sync (string)*
-----------------------------> /syncResp (string)*
-----------------------------> /syncRespFollowUp (string)*
----------------------> /packetCount*
-----------------------------> /announce (integer)*
-----------------------------> /delayReq (integer)*
-----------------------------> /delayResp (integer)*
-----------------------------> /followUp (integer)*
-----------------------------> /management (integer)*
-----------------------------> /monitoringReq (integer)*
-----------------------------> /monitoringResp (integer)*
-----------------------------> /pDelayReq (integer)*
-----------------------------> /pDelayResp (integer)*
-----------------------------> /pDelayRespFollowUp (integer)*
-----------------------------> /signalling (integer)*
-----------------------------> /sync (integer)*
-----------------------------> /syncResp (integer)*
-----------------------------> /syncRespFollowUp (integer)*
-----------------------------> /total (integer)*
```

* Directly referenceable endpoint

```
-> /api/current*
---------> /instances*
-----------------> /[sequential index number]*
----------------------> /grandmasterQuality*
------------------------------> /clockAccuracy (string)
------------------------------> /clockAccuracyText (string)
------------------------------> /clockAccuracyValue (integer)
------------------------------> /clockClass (integer)
------------------------------> /clockID (string)
------------------------------> /clockVariance (integer)
------------------------------> /priority1 (integer)
------------------------------> /priority2 (integer)
------------------------------> /valid (boolean)
----------------------> /localQuality*
------------------------------> /clockAccuracy (string)
------------------------------> /clockAccuracyText (string)
------------------------------> /clockAccuracyValue (integer)
------------------------------> /clockClass (integer)
------------------------------> /clockID (string)
------------------------------> /clockVariance (integer)
------------------------------> /priority1 (integer)
------------------------------> /priority2 (integer)
------------------------------> /valid (boolean)
----------------------> /latestTimestamp*
------------------------------> /nanoseconds (integer)
------------------------------> /seconds (integer)
------------------------------> /value (string)
```

* Directly referenceable endpoint

```
-> /api/current*
---------> /instances*
-----------------> /[sequential index number]*
----------------------> /statistics*
-----------------------------> /cto*
-----------------------------------> /[sequential index number]
-----------------------------------------> /time
-----------------------------------------> /value
-----------------------------> /ctoDelta*
-----------------------------------> /[sequential index number]
-----------------------------------------> /time
-----------------------------------------> /value
-----------------------------> /measuredDelay*
-----------------------------------> /[sequential index number]
-----------------------------------------> /time
-----------------------------------------> /value
-----------------------------> /measuredDelayDelta*
-----------------------------------> /[sequential index number]
-----------------------------------------> /time
-----------------------------------------> /value
-----------------------------> /measuredOffset*
-----------------------------------> /[sequential index number]
-----------------------------------------> /time
-----------------------------------------> /value
-----------------------------> /measuredOffsetDelta*
-----------------------------------> /[sequential index number]
-----------------------------------------> /time
-----------------------------------------> /value
-----------------------------> /reportedDelay*
-----------------------------------> /[sequential index number]
-----------------------------------------> /time
-----------------------------------------> /value
-----------------------------> /reportedDelayDelta*
-----------------------------------> /[sequential index number]
-----------------------------------------> /time
-----------------------------------------> /value
-----------------------------> /reportedOffset*
-----------------------------------> /[sequential index number]
-----------------------------------------> /time
-----------------------------------------> /value
-----------------------------> /reportedOffsetDelta*
-----------------------------------> /[sequential index number]
-----------------------------------------> /time
-----------------------------------------> /value
```

* Directly referenceable endpoint

```
-> /api/current*
---------> /packets*
-----------------> /latest (integer)*
-----------------> /oldest (integer)*
-----------------> /count*
----------------------> /announce (integer)*
----------------------> /delayReq (integer)*
----------------------> /delayResp (integer)*
----------------------> /followUp (integer)*
----------------------> /management (integer)*
----------------------> /monitoringReq (integer)*
----------------------> /monitoringResp (integer)*
----------------------> /pDelayReq (integer)*
----------------------> /pDelayResp (integer)*
----------------------> /pDelayRespFollowUp (integer)*
----------------------> /signalling (integer)*
----------------------> /sync (integer)*
----------------------> /v1 (integer)*
----------------------> /v2 (integer)*
----------------------> /v2_0 (integer)*
----------------------> /v2_1 (integer)*
----------------------> /total (integer)*
----------------------> /inStore (integer)*
----------------------> /bytes (integer)*
-----------------> /perSecond*
----------------------> /announce (float)*
----------------------> /delayReq (float)*
----------------------> /delayResp (float)*
----------------------> /followUp (float)*
----------------------> /management (float)*
----------------------> /monitoringReq (float)*
----------------------> /monitoringResp (float)*
----------------------> /pDelayReq (float)*
----------------------> /pDelayResp (float)*
----------------------> /pDelayRespFollowUp (float)*
----------------------> /signalling (float)*
----------------------> /sync (float)*
----------------------> /syncResp (float)*
----------------------> /syncRespFollowUp (float)*
----------------------> /time (integer)*
----------------------> /total (float)*
----------------------> /v1 (float)*
----------------------> /v2 (float)*
----------------------> /v2_0 (float)*
----------------------> /v2_1 (float)*
----------------------> /bytes (float)*
```

* Directly referenceable endpoint

```
-> /api/current*
---------> /scopes*
-----------------> /[sequential index number]*
----------------------> /_index (integer)
----------------------> /id (integer)*
----------------------> /domain (integer)*
----------------------> /protocol (string)*
----------------------> /version (string)*
----------------------> /compatibility (boolean)*
----------------------> /vlanID (string)*
----------------------> /segmentID (integer)*
----------------------> /_links*
-----------------------------> /self (string)
----------------------> /packetCount*
-----------------------------> /announce (integer)*
-----------------------------> /delayReq (integer)*
-----------------------------> /delayResp (integer)*
-----------------------------> /followUp (integer)*
-----------------------------> /management (integer)*
-----------------------------> /monitoringReq (integer)*
-----------------------------> /monitoringResp (integer)*
-----------------------------> /pDelayReq (integer)*
-----------------------------> /pDelayResp (integer)*
-----------------------------> /pDelayRespFollowUp (integer)*
-----------------------------> /signalling (integer)*
-----------------------------> /sync (integer)*
-----------------------------> /syncResp (integer)*
-----------------------------> /syncRespFollowUp (integer)*
-----------------------------> /inStore (integer)*
-----------------------------> /total (integer)*
---------> /segments*
-----------------> /[sequential index number]*
----------------------> /_index (integer)
----------------------> /id (integer)*
----------------------> /name (string)*
----------------------> /description (string)*
----------------------> /_links*
-----------------------------> /self (string)
```

* Directly referenceable endpoint

13.4.1 /api/current/devices/[_id]

`/_index`	*integer*	The sequentially assigned index value for the device.
`/alias`	*string*	The alias for the device as declared by the device via management messages.
`/alternateVendor`	*string*	The vendor name for the device as declared by the device via management messages.
`/clockID`	*string*	The PTP clock ID of the device.
`/firmwareRevision`	*string*	The firmware revision of the device as declared by the device via management messages.
`/hardwareRevision`	*string*	The hardware revision of the device as declared by the device via management messages.
`/softwareRevision`	*string*	The software revision of the device as declared by the device via management messages.
`/id`	*integer*	The unique ID of the device assigned by PTP Track Hound v2.
`/instances`	*integer array*	An array containing the list of instances associated with this device. These values correspond to `/api/current/instances/*/id`.
`/location`	*string*	The location of the device as declared by the device via management messages.
`/modelName`	*string*	The model name of the device as declared by the device via management messages.
`/ports`	*integer array*	An array containing the list of PTP port IDs of the instances associated with this device.
`/type`	*string*	The type of device. Valid values: `Grandmaster Clock`, `Receiver Clock`, `Ordinary Clock`, `Boundary Clock`, `Transparent Clock`, `Management Node`, `PTP Track Hound`, `Unknown`
`/vendor`	*string*	The vendor of the device as declared by the device via the IEEE OUI.

13.4.1.1 /api/current/devices/[_id]/_links

`/image`	*string*	The image set for the device to be displayed in the devices map. This is relative to the file server path `/api/config/api/fileServer/directory`.
`/self`	*string*	The API endpoint to the device itself under `/api/current/devices`.
`/vendorImage`	*string*	The image set for the device vendor. This is relative to the file server path `/api/config/api/fileServer/directory`.

13.4.1.2 /api/current/devices/[_id]/custom

`/alias`	*string*	The manually set alias for the device. Overrides the automatically acquired alias.
`/firmwareRevision`	*string*	The manually set firmware revision for the device. Overrides the automatically acquired firmware revision.
`/hardwareRevision`	*string*	The manually set hardware revision for the device. Overrides the automatically acquired hardware revision.
`/softwareRevision`	*string*	The manually set software revision for the device. Overrides the automatically acquired software revision.
`/imagePath`	*string*	The manually selected image for the device. Overrides the automatically acquired image. This value relates to the file server link as provided under `/api/files`.
`/location`	*string*	The manually set physical location for the device. Overrides the automatically acquired physical location.
`/modelName`	*string*	The manually set model name for the device. Overrides the automatically acquired model name.
`/vendor`	*string*	The manually set vendor for the device. Overrides the automatically acquired vendor name.

13.4.2 /api/current/instances/[_id]

`/_index`	*integer*	The sequentially assigned index value for the instance.
`/address`	*string*	The IPv4, IPv6, or MAC address of the instance.
`/announceReceiptTimeout`	*integer*	The number of announce messages set by the instance that a slave must miss before a timeout is declared for a clock relationship. This value is declared by the instance via management messages.
`/color`	*string*	The color used for the rendering of graphs for this instance. Is provided in RGB hexadecimal format (e.g., `#ff0000` for red).
`/ctoAbs`	*integer*	The current offset in nanoseconds between the PTP timestamp and the time of the local clock. This value is *unsigned* and provides no indication of the direction of the offset.
`/ctoCount`	*integer*	The number of Capture Time Offset data points currently recorded for this instance. This is limited by the value set at `/api/config/statistics/maxEntries`.
`/ctoDelta`	*integer*	The most recent Capture Time Offset delta value (i.e., the difference between the last two offset values) in nanoseconds.
`/ctoHwTimestamp`	*boolean*	Specifies if the local clock timestamps for Capture Time Offsets are generated by a hardware timestamper at NIC level.
`/ctoLatest`	*integer*	The host system timestamp (Unix timestamp) of the last Capture Time Offset measurement.

/ctoNsPrecision	*boolean*	Specifies whether the Capture Time Offsets are measurable in nanoseconds.
/ctoReal	*integer*	The current offset between the PTP timestamp and the time of the local clock as a PTP timestamp. This value is *signed* and therefore indicates the direction of the offset.
/delayMechanism	*string*	The delay mechanism used by the instance for measuring propagation delays. Valid values: `E2E`, `P2P`
/device	*integer*	The device ID (`/api/current/devices/*/id`) to which this instance is bound.
/frequencyTraceable	*boolean*	Specifies whether the frequency signal can be traced back to its primary reference.
/grandmaster	*integer*	Specifies the instance ID (`/api/current/instances/*/id`) of this instance's Grandmaster. If the instance is itself a Grandmaster, it will refer to its own instance (and by definition will be identical to `/id` below).
/grandmasterDevice	*integer*	Specifies the device ID (`/api/current/devices/*/id`) of this instance's Grandmaster. If the instance is itself a Grandmaster, it will refer to its own device (and by definition will be identical to `/device` above).
/id	*integer*	The ID set by the PTP Track Hound v2 capture service for this instance.
/illegalMaster	*boolean*	Specifies if this instance is an illegal master.
/leap59	*boolean*	Specifies if the removal of a leap second is scheduled for this instance.
/leap61	*boolean*	Specifies if the insertion of a leap second is scheduled for this instance.
/measuredDelayAbs	*integer*	The most recent path delay measurement as performed using NetSync Monitor. This value is *unsigned* and provides no indication of the direction of the delay.
/measuredDelayCount	*integer*	The number of stored path delay measurements as performed using NetSync Monitor. This is limited by the value set at `/api/config/statistics/maxEntries`.
/measuredDelayDelta	*integer*	The difference between the last two path delay measurements as performed using NetSync Monitor.
/measuredDelayHwTimestamp	*boolean*	Specifies if the local clock timestamps for NetSync Monitor path delay measurements are generated by a hardware timestamper at NIC level.
/measuredDelayLatest	*integer*	The host system Unix timestamp of the latest NetSync Monitor path delay measurement data point recorded for this instance.
/measuredDelayReal	*integer*	The most recent path delay measurement as performed using NetSync Monitor. This value is *signed* and therefore indicates the direction of the delay.

`/measuredOffsetAbs`	*integer*	The most recent clock offset measurement as performed using NetSync Monitor. This value is *unsigned* and provides no indication of the direction of the offset.
`/measuredOffsetCount`	*integer*	The number of stored clock offset measurements as performed using NetSync Monitor. This is limited by the value set at `/api/config/statistics/maxEntries`.
`/measuredOffsetDelta`	*integer*	The difference between the last two clock offset measurements performed using NetSync Monitor.
`/measuredOffsetHwTimestamp`	*boolean*	Specifies if the local clock timestamps for NetSync Monitor clock offset measurements are generated by a hardware timestamper at NIC level.
`/measuredOffsetLatest`	*integer*	The host system Unix timestamp of the latest NetSync Monitor clock offset measurement data point recorded for this instance.
`/measuredOffsetReal`	*integer*	The most recent clock offset measurement as performed using NetSync Monitor. This value is *signed* and therefore indicates the direction of the offset.
`/parent`	*integer*	Specifies the ID of the parent instance (`/api/current/instances/*/id`) of this instance. If this instance is a top-level instance, this value will be `null`.
`/parentDevice`	*integer*	Specifies the ID of the parent instance's device (`/api/current/devices/*/id`) for this instance. If this instance is a top-level instance, this value will be `null`.
`/port`	*integer*	The port number of the instance; this is conventionally appended to the clockID.
`/reportedDelayAbs`	*integer*	The most recent path delay data point received via Management Messages. This value is *unsigned* and provides no indication of the direction of the delay.
`/reportedDelayCount`	*integer*	The number of stored path delay data points received via Management Messages. This is limited by the value set at `/api/config/statistics/maxEntries`.
`/reportedDelayDelta`	*integer*	The difference between the last two path delay data points received via Management Messages.
`/reportedDelayLatest`	*integer*	The host system Unix timestamp of the latest path delay measurement data point recorded for this instance on the basis of Management Messages.
`/reportedDelayReal`	*integer*	The most recent path delay data point received via Management Messages. This value is *signed* and therefore indicates the direction of the delay.
`/reportedOffsetAbs`	*integer*	The most recent clock offset data point received via Management Messages. This value is *unsigned* and provides no indication of the direction of the offset.
`/reportedOffsetCount`	*integer*	The number of stored clock offset data points received via Management Messages. This is limited by the value set at `/api/config/statistics/maxEntries`.

`/reportedOffsetDelta`	*integer*	The difference between the last two clock offset data points received via Management Messages.
`/reportedOffsetLatest`	*integer*	The host system Unix timestamp of the latest clock offset measurement data point recorded for this instance on the basis of Management Messages.
`/reportedOffsetReal`	*integer*	The most recent clock offset data point received via Management Messages. This value is *signed* and therefore indicates the direction of the delay.
`/scope`	*integer*	The scope ID that this instance has been assigned to by PTP Track Hound v2.
`/sequenceIDAnnounce`	*integer*	The sequence ID of the last Announce message to be received from this instance.
`/sequenceIDRequest`	*integer*	The sequence ID of the last (Peer) Delay Request message to be received from this instance.
`/sequenceIDSync`	*integer*	The sequence ID of the last Sync message to be received from this instance.
`/state`	*string*	The port state of this instance. The strings here reference the terminology defined at `/api/config/terminology/states/*/value`
`/stepsRemoved`	*integer*	The number of steps that this instance is removed from its Grandmaster. Will be `0` if the instance is itself the Grandmaster.
`/timeSource`	*string*	The primary time reference for this instance as provided via *Announce* or *Management* messages, if known. Valid values: `Atomic Clock`, `GPS`, `Terrestrial Radio`, `PTP`, `NTP`, `Hand-Set`, `Other`, `Internal Oscillator`, `Unknown`
`/timeTraceable`	*boolean*	Specifies whether the time can be traced back to its primary reference.
`/timescale`	*boolean*	If `true`, this instance is using the PTP timescale. If `false`, this instance is using an arbitrary timescale.
`/timestampMechanism`	*string*	Specifies the timestamping mechanism of this instance. Valid values: `One-Step`, `Two-Step`
`/utcOffset`	*integer*	Specifies the offset between PTP time and UTC time in seconds.

13.4.2.1 /api/current/instances/[_id]/custom

`/address`	*string*	Specifies the custom alias used for this instance instead of the hostname or address.
`/color`	*string*	Specifies the custom color used for this instance for the rendering of graphs in RGB hexadecimal format (e.g., `#ff0000` for red).

13.4.2.2 /api/current/instances/[_id]/_links

`/device`	*string*	The API endpoint to the device on which the instance is running.
`/grandmaster`	*string*	The API endpoint to the instance that serves as Grandmaster to this instance. If the instance is itself a Grandmaster, it will refer to its own instance.
`/grandmasterdevice`	*string*	The API endpoint to the device on which the instance serving as Grandmaster to this instance is running. If the instance is itself a Grandmaster, it will refer to its own instance.
`/parent`	*integer*	The API endpoint to the parent instance of this instance. This value will only exist if the instance is not a top-level instance.
`/parentDevice`	*integer*	The API endpoint to the parent device for this instance. This value will only exist if the instance is not a top-level instance.
`/scope`	*string*	The API endpoint of the scope to which this instance has been assigned.
`/self`	*string*	This instance's own API endpoint as a link.

13.4.2.3 /api/current/instances/[_id]/intervals

All resources under /api/current/instances/*/intervals support the same valid values:

`128/s, 64/s, 32/s, 16/s, 8/s, 4/s, 2/s, 1/s, 1/2s, 1/4s, 1/8s, 1/16s, 1/32s, 1/64s, 1/128s, n/a`

`/announce`	*string*	Specifies the *Announce* message interval of the instance.
`/delayReq`	*string*	Specifies the *Delay Request* message interval of the instance.
`/delayResp`	*string*	Specifies the *Delay Response* message interval of the instance.
`/followUp`	*string*	Specifies the *Follow Up* message interval of the instance.
`/management`	*string*	Specifies the *Management* message interval of the instance.
`/monitoringReq`	*string*	Specifies the *Monitoring Request* message interval of the instance. This relates to NetSync Monitor measurements and will read `n/a` if the instance does not support NetSync Monitor or does not report such an interval.
`/monitoringResp`	*string*	Specifies the *Monitoring Response* message interval of the instance. This relates to NetSync Monitor measurements and will read `n/a` if the instance does not support NetSync Monitor or does not report such an interval.
`/pDelayReq`	*string*	Specifies the *Peer Delay Request* message interval of the instance.
`/pDelayResp`	*string*	Specifies the *Peer Delay Response* message interval of the instance.
`/pDelayRespFollowUp`	*string*	Specifies the *Peer Delay Response Follow Up* message interval of the instance.
`/signalling`	*string*	Specifies the *Signaling* message interval of the instance.

`/sync`	*string*	Specifies the *Sync* message interval of the instance.
`/syncResp`	*string*	Specifies the *Sync Response* message interval of the instance. Relates to *flashPTP* nodes.
`/syncRespFollowUp`	*string*	Specifies the *Sync Response Follow Up* message interval of the instance. Relates to *flashPTP* nodes.

13.4.2.4 /api/current/instances/[_id]/latestTimestamp

`/seconds`	*integer*	The number of seconds in the latest timestamp recorded for this instance in PTP time (whole seconds).
`/nanoseconds`	*integer*	The number of nanoseconds in the latest timestamp recorded for this instance in PTP time.
`/value`	*string*	The latest timestamp as an ISO 8601-compliant string in the format: `yyyy-MM-dd'T'HH:mm:ss.SSSSSSSSS`

13.4.2.5 /api/current/instances/[_id]/packetCount

`/announce`	*integer*	Specifies the total number of *Announce* messages counted for this instance since last erasure.
`/delayReq`	*integer*	Specifies the total number of *Delay Request* messages counted for this instance since last erasure.
`/delayResp`	*integer*	Specifies the total number of *Delay Response* messages counted for this instance since last erasure.
`/followUp`	*integer*	Specifies the total number of *Follow Up* messages counted for this instance since last erasure.
`/management`	*integer*	Specifies the total number of *Management* messages counted for this instance since last erasure.
`/monitoringReq`	*integer*	Specifies the total number of *Monitoring Request* messages counted for this instance since last erasure.
`/monitoringResp`	*integer*	Specifies the total number of *Monitoring Response* messages counted for this instance since last erasure.
`/pDelayReq`	*integer*	Specifies the total number of *Peer Delay Request* messages counted for this instance since last erasure.
`/pDelayResp`	*integer*	Specifies the total number of *Peer Delay Response* messages counted for this instance since last erasure.
`/pDelayRespFollowUp`	*integer*	Specifies the total number of *Peer Delay Response Follow Up* messages counted for this instance since last erasure.
`/signalling`	*integer*	Specifies the total number of *Signaling* messages counted for this instance since last erasure.
`/sync`	*integer*	Specifies the total number of *Sync* messages counted for this instance since last erasure.
`/syncResp`	*integer*	Specifies the total number of *Sync Response* messages counted for this instance since last erasure.

`/syncRespFollowUp`	*integer*	Specifies the total number of *Sync Response Follow Up* messages counted for this instance since last erasure.
`/total`	*integer*	Specifies the total number of PTP messages counted for this instance since last erasure.

13.4.2.6 /api/current/instances/[_id]/grandmasterQuality

`/clockAccuracy`	*string*	The accuracy of the Grandmaster according to the PTP dataset, followed by the IEEE 1588 accuracy state code as a hexadecimal value in parentheses.
`/clockAccuracyText`	*string*	The accuracy of the Grandmaster according to the PTP dataset, without the IEEE 1588 accuracy state code.
`/clockAccuracyValue`	*integer*	The accuracy of the Grandmaster according to the PTP dataset, as the IEEE 1588 accuracy state in decimal.
`/clockClass`	*integer*	The Clock Class of the Grandmaster according to the PTP dataset.
`/clockID`	*string*	The Clock ID of the Grandmaster according to the PTP dataset.
`/clockVariance`	*string*	The Clock Variance of the Grandmaster according to the PTP dataset.
`/priority1`	*integer*	The Priority 1 value of the Grandmaster according to the PTP dataset.
`/priority2`	*integer*	The Priority 2 value of the Grandmaster according to the PTP dataset.
`/valid`	*boolean*	Specifies whether the dataset of the Grandmaster is valid.

13.4.2.7 /api/current/instances/[_id]/localQuality

`/clockAccuracy`	*string*	The accuracy of the clock according to the PTP dataset, followed by the IEEE 1588 accuracy state code as a hexadecimal value in parentheses.
`/clockAccuracyText`	*string*	The accuracy of the clock according to the PTP dataset, without the IEEE 1588 accuracy state code.
`/clockAccuracyValue`	*integer*	The accuracy of the clock according to the PTP dataset, as the IEEE 1588 accuracy state code in decimal.
`/clockClass`	*integer*	The Clock Class of the clock according to the PTP dataset.
`/clockID`	*string*	The Clock ID of the clock according to the PTP dataset.
`/clockVariance`	*string*	The Clock Variance of the clock according to the PTP dataset.
`/priority1`	*integer*	The Priority 1 value of the clock according to the PTP dataset.
`/priority2`	*integer*	The Priority 2 value of the clock according to the PTP dataset.
`/valid`	*boolean*	Specifies whether the dataset of the clock is valid.

13.4.2.8 /api/current/instances/[_id]/statistics

Due to the large size of the JSON output from this endpoint, the analysis statistics for each instance are not provided at any higher-level endpoint. They are only available by calling the endpoint `/api/current/instances/[_id]/statistics` directly. Individual statistic types are available by referencing the statistic type in the API endpoint:

/cto/[_id]

`/time`	*integer*	The host Unix timestamp of the Capture Time Offset measurement.
`/value`	*integer*	The Capture Time Offset for the instance, as recorded at the time: `/api/current/instances/[_id]/statistics/cto/[_id]/time`

/ctoDelta/[_id]

`/time`	*integer*	The host Unix timestamp of the Capture Time Offset delta calculation.
`/value`	*integer*	The difference between this Capture Time Offset value and the previous recorded value for the instance, as recorded at the time: `/api/current/instances/[_id]/statistics/ctoDelta/[_id]/time`

/measuredDelay/[_id]

`/time`	*integer*	The host Unix timestamp of the NetSync Monitor path delay measurement.
`/value`	*integer*	The NetSync Monitor path delay measurement for the instance, as recorded at the time: `/api/current/instances/[_id]/statistics/measuredDelay/[_id]/time`

/measuredDelayDelta/[_id]

`/time`	*integer*	The host Unix timestamp of the NetSync Monitor path delay delta calculation.
`/value`	*integer*	The difference between this NetSync Monitor path delay measurement and the previous recorded value for the instance, as recorded at the time: `/api/current/instances/[_id]/statistics/measuredOffsetDelta/[_id]/time`

/measuredOffset/[_id]

`/time`	*integer*	The host Unix timestamp of the NetSync Monitor clock offset measurement.
`/value`	*integer*	The NetSync Monitor clock offset measurement for the instance, as recorded at the time: `/api/current/instances/[_id]/statistics/measuredOffset/[_id]/time`

/measuredOffsetDelta/[_id]

`/time`	*integer*	The host Unix timestamp of the NetSync Monitor clock offset delta calculation.
`/value`	*integer*	The difference between this NetSync Monitor clock offset measurement and the previous recorded value for the instance, as recorded at the time: `/api/current/instances/[_id]/statistics/measuredOffsetDelta/[_id]/time`

/reportedDelay/[_id]

`/time`	*integer*	The host Unix timestamp of the path delay for the instance, as reported via PTP Management Messages.
`/value`	*integer*	The path delay for the instance reported via PTP Management Messages, as recorded at the time: `/api/current/instances/[_id]/statistics/reportedDelay/[_id]/time`

/reportedDelayDelta/[_id]

`/time`	*integer*	The host Unix timestamp of the Management Message path delay delta calculation.
`/value`	*integer*	The difference between this path delay value reported via PTP Management Messages and the previous reported value for the instance, as recorded at the time: `/api/current/instances/[_id]/statistics/reportedDelayDelta/[_id]/time`

/reportedOffset/[_id]

`/time`	*integer*	The host Unix timestamp of the clock offset for the instance, as reported via PTP Management Messages.
`/value`	*integer*	The clock offset for the instance reported via PTP Management Messages, as recorded at the time: `/api/current/instances/[_id]/statistics/reportedDelay/[_id]/time`

/reportedOffsetDelta/[_id]

`/time`	*integer*	The host Unix timestamp of the Management Message clock offset delta calculation.
`/value`	*integer*	The difference between this clock offset value reported via PTP Management Messages and the previous reported value for the instance, as recorded at the time: `/api/current/instances/[_id]/statistics/reportedOffsetDelta/[_id]/time`

13.4.2.9 /api/current/instances/[_id]/captureTimeOffsetStatistics (Disabled)

Information:

The API route `/api/current/instances/[_id]/captureTimeOffsetStatistics` introduced in PTP Track Hound v2.0.5 no longer exists as of PTP Track Hound v2.1.0.

Capture Time Offset statistics for each instance can now be read via the route:

`/api/current/instances/[_id]/statistics/cto`

Refer to Chapter 13.4.2.8 for more information.

13.4.3 /api/current/packets

`/latest`	*integer*	Specifies the ID number of the latest PTP packet to be captured.
`/oldest`	*integer*	Specifies the ID number of the oldest PTP packet currently in the capture log.

13.4.3.1 /api/current/packets/count

`/announce`	*integer*	Specifies the total number of *Announce* messages counted since last erasure.
`/delayReq`	*integer*	Specifies the total number of *Delay Request* messages counted since last erasure.
`/delayResp`	*integer*	Specifies the total number of *Delay Response* messages counted since last erasure.
`/followUp`	*integer*	Specifies the total number of *Follow Up* messages counted since last erasure.
`/management`	*integer*	Specifies the total number of *Management* messages counted since last erasure.
`/monitoringReq`	*integer*	Specifies the total number of *Monitoring Request* messages counted since last erasure.
`/monitoringResp`	*integer*	Specifies the total number of *Monitoring Response* messages counted since last erasure.
`/pDelayReq`	*integer*	Specifies the total number of *Peer Delay Request* messages counted since last erasure.
`/pDelayResp`	*integer*	Specifies the total number of *Peer Delay Response* messages counted since last erasure.
`/pDelayRespFollowUp`	*integer*	Specifies the total number of *Peer Delay Response Follow Up* messages counted since last erasure.
`/signalling`	*integer*	Specifies the total number of *Signaling* messages counted since last erasure.

`/sync`	*integer*	Specifies the total number of *Sync* messages counted since last erasure.
`/v1`	*integer*	Specifies the total number of incoming PTPv1 messages counted since last erasure.
`/v2`	*integer*	Specifies the total number of incoming PTPv2 and PTPv2.1 messages counted since last erasure.
`/v2_0`	*integer*	Specifies the total number of incoming PTPv2 (not PTPv2.1) messages counted since last erasure.
`/v2_1`	*integer*	Specifies the total number of incoming PTPv2.1 messages counted since last erasure.
`/total`	*integer*	Specifies the total number of PTP messages counted since last erasure.
`/inStore`	*integer*	Specifies the total number of PTP messages currently held in the capture log.
`/bytes`	*integer*	Specifies the total content of PTP packets counted since last erasure in bytes.

13.4.3.2 /api/current/packets/perSecond

For more information on how these per-second values are calculated, please refer to Chapter 8, "`Traffic`".

`/announce`	*float*	Specifies the current number of *Announce* messages received per second.
`/delayReq`	*float*	Specifies the current number of *Delay Request* messages received per second.
`/delayResp`	*float*	Specifies the current number of *Delay Response* messages received per second.
`/followUp`	*float*	Specifies the current number of *Follow Up* messages received per second.
`/management`	*float*	Specifies the current number of *Management* messages received per second.
`/monitoringReq`	*float*	Specifies the current number of *Monitoring Request* messages received per second.
`/monitoringResp`	*float*	Specifies the current number of *Monitoring Response* messages received per second.
`/pDelayReq`	*float*	Specifies the current number of *Peer Delay Request* messages received per second.
`/pDelayResp`	*float*	Specifies the current number of *Peer Delay Response* messages received per second.
`/pDelayRespFollowUp`	*float*	Specifies the current number of *Peer Delay Response Follow Up* messages received per second.
`/signalling`	*float*	Specifies the current number of *Signaling* messages received per second.

`/sync`	*float*	Specifies the current number of *Sync* messages received per second.
`/syncResp`	*float*	Specifies the current number of *Sync Response* messages received per second.
`/syncRespFollowUp`	*float*	Specifies the current number of *Sync Response Follow Up* messages received per second.
`/time`	*float*	The time at which the last per-second values were calculated as a PTP timestamp.
`/total`	*float*	Specifies the current number of PTP messages in general received per second.
`/v1`	*float*	Specifies the current number of PTPv1 messages received per second.
`/v2`	*float*	Specifies the current number of PTPv2 and PTPv2.1 messages received per second.
`/v2_0`	*float*	Specifies the current number of PTPv2 (not PTPv2.1) messages received per second.
`/v2_1`	*float*	Specifies the current number of PTPv2.1 messages received per second.
`/bytes`	*float*	Specifies the current data rate of incoming PTP packets per second in bytes.

13.4.4 /api/current/scopes/[_id]

`/_index`	*integer*	The sequentially assigned index value for the scope.
`/id`	*integer*	The unique ID of the scope assigned by PTP Track Hound v2.
`/domain`	*integer* or *string*	The common domain or subdomain shared by all instances in this scope. The datatype is dependent on the scope type. If the scope is a PTPv1 scope, this will be a string. If it is a PTPv2 or PTPv2.1 scope, it will be an integer.
`/protocol`	*string*	The common network protocol employed by all instances in this scope. Valid values: `IPv4`, `IPv6`, `IEEE 802.3`
`/version`	*string*	The common PTP version employed by all instances in this scope. Valid values: `PTPv1`, `PTPv2`, `PTPv2.1`
`/compatibility`	*boolean*	If `true`, this is a formerly PTPv2 scope that has been changed to PTPv2.1 due to the presence of at least one PTPv2.1 instance but still contains PTPv2 instances. If `false`, this is a PTPv2.1 scope that only contains PTPv2.1 instances. Accordingly, this resource does not appear in non-PTPv2.1 scopes.
`/vlanID`	*string*	The common VLAN tag employed by all instances in this scope. If VLAN tagging is not used, this will be `none`.
`/segmentID`	*integer*	The manually defined segment ID to which all instances in this scope belong.

13.4.4.1 /api/current/scopes/[_id]/_links

`/self`	*string*	This scope's own API endpoint as a link.

13.4.4.2 /api/current/scopes/[_id]/packetCount

`/announce`	*integer*	Specifies the total number of *Announce* messages counted for this scope since last erasure.
`/delayReq`	*integer*	Specifies the total number of *Delay Request* messages counted for this scope since last erasure.
`/delayResp`	*integer*	Specifies the total number of *Delay Response* messages counted for this scope since last erasure.
`/followUp`	*integer*	Specifies the total number of *Follow Up* messages counted for this scope since last erasure.
`/management`	*integer*	Specifies the total number of *Management* messages counted for this scope since last erasure.
`/monitoringReq`	*integer*	Specifies the total number of *Monitoring Request* messages counted for this scope since last erasure.
`/monitoringResp`	*integer*	Specifies the total number of *Monitoring Response* messages counted for this scope since last erasure.
`/pDelayReq`	*integer*	Specifies the total number of *Peer Delay Request* messages counted for this scope since last erasure.
`/pDelayResp`	*integer*	Specifies the total number of *Peer Delay Response* messages counted for this scope since last erasure.
`/pDelayRespFollowUp`	*integer*	Specifies the total number of *Peer Delay Response Follow Up* messages counted for this scope since last erasure.
`/signalling`	*integer*	Specifies the total number of *Signaling* messages counted for this scope since last erasure.
`/sync`	*integer*	Specifies the total number of *Sync* messages counted for this scope since last erasure.
`/syncResp`	*integer*	Specifies the total number of *Sync Response* messages counted for this scope since last erasure.
`/syncRespFollowUp`	*integer*	Specifies the current number of *Sync Response Follow Up* messages counted for this scope since last erasure.
`/inStore`	*integer*	Specifies the total number of PTP messages currently held in the capture log for this scope.
`/total`	*integer*	Specifies the total number of PTP messages counted for this scope since last erasure.

13.4.5 /api/current/segments/[_id]

`/_index`	*integer*	The sequentially assigned index value for the segment.
`/id`	*integer*	The unique ID of the segment assigned by PTP Track Hound v2.
`/name`	*string*	The manually entered name of the segment that appears in the Web Interface as the heading.
`/description`	*string*	The manually entered description for the segment that appears in the Web Interface beneath the heading.

13.4.5.1 /api/current/segments/[_id]/_links

`/self`	*string*	This segment's own API endpoint as a link.

13.5 /api/events

The `events` resources provide information about logged events in PTP Track Hound v2.

```
-> /api/events⋆
---------> /count (integer)
---------> /[sequential index number]⋆#
------------------> /_index (integer)
------------------> /device (integer)
------------------> /event (string)
------------------> /brief (string)
------------------> /details (string)
------------------> /instance (integer)
------------------> /id (integer)
------------------> /packet (integer)
------------------> /port (integer)
------------------> /scope (integer)
------------------> /severity (string)
------------------> /timestamp (string)
------------------> /time
-----------------------> /seconds (integer)
-----------------------> /nanoseconds (integer)
---------> /latest+
------------------> /device (integer)
------------------> /event (string)
------------------> /brief (string)
------------------> /details (string)
------------------> /instance (integer)
------------------> /id (integer)
------------------> /packet (integer)
------------------> /port (integer)
------------------> /scope (integer)
------------------> /severity (string)
------------------> /timestamp (string)
------------------> /time
-----------------------> /seconds (integer)
-----------------------> /nanoseconds (integer)
```

⋆ Directly referenceable endpoint.

+ Only referenceable from the root endpoint `/api/`.

# The full list of events is only returned via the root endpoint `/api/`.

```
---------> /oldest+
-----------------> /device (integer)
-----------------> /event (string)
-----------------> /brief (string)
-----------------> /details (string)
-----------------> /instance (integer)
-----------------> /id (integer)
-----------------> /packet (integer)
-----------------> /port (integer)
-----------------> /scope (integer)
-----------------> /severity (string)
-----------------> /timestamp (string)
-----------------> /time
----------------------> /seconds (integer)
----------------------> /nanoseconds (integer)
```

+ Only referenceable from the root endpoint `/api/`.

`/count`	*integer*	The total number of events currently in store.

/[_id]

`/_index`	*integer*	The sequential index number assigned in the resource level above.
`/device`	*integer*	The device ID to which the event relates.
`/event`	*string*	The type of event. Valid values: `Capture Started`, `Capture Stopped`, `Scope Detected`, `Device Detected`, `Port Detected`, `Instance Detected`, `Grandmaster Changeover`, `Port State Changed`, `Local Quality Changed`, `Grandmaster Quality Changed`, `Custom Alarm Triggered`, `Custom Alarm Cleared`
`/brief`	*string*	A brief description of the event.
`/details`	*string*	Where appropriate, additional details regarding the event (such as dataset changes) are provided here.
`/instance`	*integer*	The instance ID to which the event relates.
`/id`	*integer*	The ID set by the PTP Track Hound v2 capture service for this event.
`/packet`	*integer*	The packet ID to which the event relates.
`/port`	*integer*	The port ID of the device to which the event relates.
`/scope`	*integer*	The scope ID to which the event relates.
`/severity`	*string*	The severity of this event; this relates to the log levels of the various logging options.
`/timestamp`	*string*	The timestamp of the event in the format

YYYY-MM-DD'T'HH:MM:SS.SSSSSS.

13.5.1 /api/events/[_id]/time

/seconds	*integer*	The event timestamp in PTP time (whole seconds). Corresponds to the Gregorian timestamp under /api/events/*/timestamp.
/nanoseconds	*integer*	The number of nanoseconds after the last whole second in PTP time of the event. Corresponds to the Gregorian timestamp under /api/events/*/timestamp.

13.5.2 /api/events/latest

/device	*integer*	The device ID to which the latest event relates.
/event	*string*	The type of the latest event. Valid values: Capture Started, Capture Stopped, Scope Detected, Device Detected, Port Detected, Instance Detected, Grandmaster Changeover, Port State Changed, Local Quality Changed, Grandmaster Quality Changed, Custom Alarm Triggered, Custom Alarm Cleared
/brief	*string*	A brief description of the latest event.
/details	*string*	Where appropriate, additional details regarding the latest event (such as dataset changes) are provided here.
/instance	*integer*	The instance ID to which the latest event relates.
/id	*integer*	The ID set by the PTP Track Hound v2 capture service for the latest event.
/packet	*integer*	The packet ID to which the latest event relates.
/port	*integer*	The port ID of the device to which the event relates.
/scope	*integer*	The scope ID to which the latest event relates.
/severity	*string*	The severity of the latest event; this relates to the log levels of the various logging options.
/timestamp	*string*	The timestamp of the latest event in the format YYYY-MM-DD'T'HH:MM:SS.SSSSSS.

13.5.2.1 /api/events/[_id]/latest/time

/seconds	*integer*	The event timestamp in PTP time (whole seconds). Corresponds to the Gregorian timestamp under /api/events/*/timestamp.
/nanoseconds	*integer*	The number of nanoseconds after the last whole second in PTP time of the event. Corresponds to the Gregorian timestamp under /api/events/*/timestamp.

13.5.3 /api/events/oldest

`/device`	*integer*	The device ID to which the oldest event relates.
`/event`	*string*	The type of the oldest event. Valid values: `Capture Started`, `Capture Stopped`, `Scope Detected`, `Device Detected`, `Port Detected`, `Instance Detected`, `Grandmaster Changeover`, `Port State Changed`, `Local Quality Changed`, `Grandmaster Quality Changed`, `Custom Alarm Triggered`, `Custom Alarm Cleared`
`/brief`	*string*	A brief description of the oldest event.
`/details`	*string*	Where appropriate, additional details regarding the oldest event (such as dataset changes) are provided here.
`/instance`	*integer*	The instance ID to which the oldest event relates.
`/id`	*integer*	The ID set by the PTP Track Hound v2 capture service for the oldest event.
`/packet`	*integer*	The packet ID to which the oldest event relates.
`/port`	*integer*	The port ID of the device to which the event relates.
`/scope`	*integer*	The scope ID to which the oldest event relates.
`/severity`	*string*	The severity of the oldest event; this relates to the log levels of the various logging options.
`/timestamp`	*string*	The timestamp of the oldest event in the format `YYYY-MM-DD'T'HH:MM:SS.SSSSSS`.

13.5.3.1 /api/events/[_id]/oldest/time

`/seconds`	*integer*	The event timestamp in PTP time (whole seconds). Corresponds to the Gregorian timestamp under `/api/events/*/timestamp`.
`/nanoseconds`	*integer*	The number of nanoseconds after the last whole second in PTP time of the event. Corresponds to the Gregorian timestamp under `/api/events/*/timestamp`.

13.6 /api/info

The `info` resources provide information about the PTP Track Hound v2 version, the state of the capture service, the system on which it is running, and the registered license.

These resources can only be fetched as a complete JSON object from `/api/info`.

```
-> /api/info*
---------> /buildDate (string)
---------> /name (string)
---------> /platform (string)
---------> /running (boolean)
---------> /version (string)
---------> /license
-----------------> /id (string)
-----------------> /key (string)
-----------------> /type (string)
-----------------> /user (string)
-----------------> /version (string)
-----------------> /expiration (string)
```

* Directly referenceable endpoint

`/buildDate`	*string*	The build date and time of this version of PTP Track Hound v2 in the format `YYYY-MM-DD'T'HH:MM:SS`
`/name`	*string*	The name of the software. Always returns `PTP Track Hound`.
`/platform`	*string*	The name of the operating system on which PTP Track Hound v2 is running. Valid values: `Windows`, `Linux`, `macOS`
`/running`	*boolean*	Is `true` if the PTP Track Hound v2 capture service is running, or if a capture file is currently open and being analyzed.
`/version`	*string*	The PTP Track Hound v2 version.

13.6.1 /api/info/license

`/id`	*string*	The ID code of the currently registered license.
`/key`	*string*	The authentication key of the currently registered license.
`/type`	*string*	The license level of the currently registered license. While the valid values in theory are `Free`, `Capture`, `Basic`, and `Professional`, the REST API requires a Professional License to access, so that in practice, this resource will only ever be readable as `Professional` via the REST API.
`/user`	*string*	The name of the user associated with the currently registered license.
`/version`	*string*	The PTP Track Hound version to which this license applies.
`/expiration`	*string*	This resource provides the expiry date of the license in the format `YYYY-MM-DD` if the license is time-limited.

13.7 /api/memory

The `/api/memory` resources provide information about the memory usage settings and current memory usage of the PTP Track Hound v2 service.

These resources can only be fetched as a complete JSON object from `/api/memory`.

```
-> /api/memory*
---------> /chunkSize (integer)
---------> /maxBytes (integer)
---------> /used
-----------------> /data (integer)
-----------------> /model (integer)
-----------------> /total (integer)
```

* Directly referenceable endpoint

`/chunkSize`	*integer*	The currently configured size of each allocable memory chunks in bytes. Identical to `/api/config/memory/chunkSize`.
`/maxBytes`	*integer*	The currently configured maximum amount of memory in bytes usable by PTP Track Hound v2 for captured data, analysis, and modeling. Identical to `/api/config/memory/max`.

13.7.1 /api/memory/used

`/data`	*integer*	The amount of memory in bytes currently used by the PTP Track Hound v2 service for the storage of capture data.
`/model`	*integer*	The amount of memory in bytes currently used by the PTP Track Hound v2 service for data modeling.
`/total`	*integer*	The total amount of memory in bytes currently used by the PTP Track Hound v2 service for the storage of capture data and data modeling. `/api/memory/maxBytes` represents the maximum that this value can reach.

13.8 /api/network

The `/api/network` resources provide information about the network interfaces found by PTP Track Hound's packet capture service. Under Windows, this is the information gathered by Npcap.

These resources can only be fetched as a complete JSON object from `/api/network`.

```
-> /api/network*
---------> /interfaces
---------------> /[sequential index number]
---------------------> /description (string)
---------------------> /friendlyName (string)
---------------------> /hardwareTimestamps (boolean)
---------------------> /index (integer)
---------------------> /ipAddresses (string array)
---------------------> /linkState (string)
---------------------> /macAddress (string)
---------------------> /name (string)
---------------------> /timestampTypes (string)
```

* Directly referenceable endpoint

13.8.1 /api/network/interfaces/[_id]

`/index`	*integer*	The index number assigned by the operating system to this physical network interface.
`/ipAddresses`	*string array*	An array containing the IPv4 and IPv6 addresses of the virtual interfaces assigned to this physical network interface.
`/linkState`	*string*	The physical link state of this physical network interface. Valid values: `up`, `down`.
`/hardwareTimestamps`	*string*	Specifies whether the network interface supports hardware timestamping.
`/timestampTypes`	*integer*	A 6-bit bitmask value showing the timestamp types supported by this interface (according to the capture library): - **Bit 0:** Host-provided, unknown characteristics - **Bit 1:** Host-provided, low-precision, synchronized with system clock - **Bit 2:** Host-provided, high-precision, synchronized with system clock - **Bit 3:** Device-provided, synchronized with system clock - **Bit 4:** Device-provided, synchronized with system clock - **Bit 5:** Device-provided, synchronized with system clock - **Bit 6:** Host-provided, high-precision, not synchronized with system clock
`/macAddress`	*string*	The MAC address of this physical network interface.
`/name`	*string*	The name of the network interface as assigned by the packet capture service.
`/friendlyName`	*string*	The friendly name of the network interface as assigned by the operating system. Only applicable to Windows.
`/description`	*string*	The description of the network interface as assigned by the operating system. Only applicable to Windows.

13.9 /api/files

The `/api/files` resources provide information about the files currently registered on the file server.

These resources can only be fetched as a complete JSON object from `/api/files`. These resources cannot be acquired via the root endpoint `/api`.

```
-> /api/files*
--------> /capture
---------------> /[sequential index number]
------------------------> /apiPath (string)
------------------------> /filename (string)
------------------------> /localPath (string)
------------------------> /size (integer)
--------> /img
---------------> /[sequential index number]
------------------------> /apiPath (string)
------------------------> /filename (string)
------------------------> /localPath (string)
------------------------> /size (integer)
```

* Directly referenceable endpoint

13.9.1 /api/files/capture

`/apiPath`	*string*	The API endpoint referencing this capture file.
`/filename`	*string*	The filename (without the path) of this capture file.
`/localPath`	*string*	The full path to the capture file, relative to the local device on which PTP Track Hound v2 is running.
`/size`	*integer*	The size of the file in bytes.

13.9.2 /api/files/img

`/apiPath`	*string*	The API endpoint referencing this image file.
`/filename`	*string*	The filename (without the path) of this image file.
`/localPath`	*string*	The full path to the image file, relative to the local device on which PTP Track Hound v2 is running.
`/size`	*integer*	The size of the file in bytes.

14 Appendix

14.1 Operation from Command Line Interface

While PTP Track Hound v2 is primarily designed to be operated via the easy-to-use Web Interface (and, under Windows, a guided installation process), the service itself and the associated tools provide even more flexibility via parameters that are accessible when called from a command line environment.

`trackhound-service`

`trackhound-service` is the main background service for PTP Track Hound v2 and accepts the following parameters.

`-h`	`--help`		Displays usage information for `trackhound-service`.
`-c`	`--config`	`<filename>`	Specifies a JSON-formatted configuration file for PTP Track Hound v2.
`-s`	`--store`	`<filename>`	Specifies a custom JSON store file in which capture and analysis data will be stored.
`-u`	`--user`	`<username>`	Specifies the licensee name as provided in the license details. This must be used in conjunction with the `-i` and `-k` switches.
`-i`	`--id`	`<idname>`	Specifies the license ID as provided in the license details. This must be used in conjunction with the `-u` and `-k` switches.
`-k`	`--key`	`<idname>`	Specifies the license key as provided in the license details. This must be used in conjunction with the -u and -i switches.
`-e`	`--expiration`	`<yyyy-mm-dd>`	Specifies the date on which the license expires if applicable. This must match the expiration date of the license key itself.
`-f`	`--fork`		Forks the process into the background so that the console window can be closed without terminating the capture service.
`-l`	`--license`	`<filename>`	Specifies a license file name containing the license details. Such a license file can be stored as a JSON file formatted as follows:

```
{ "user":  "<licensee-name>", "id":  "<license-id>", "key":
"<license-key>" }
```

trackhound-config

trackhound-config is a guided setup wizard intended to aid with the initial configuration of PTP Track Hound v2. It accepts the following parameters.

-h	--help		Displays usage information for trackhound-config.
-c	--cert	<filename>	Specifies an SSL/TLS key file to be used for HTTPS. This must be specified in conjunction with the -k switch. If an SSL/TLS certificate and private key are specified as command line parameters, the corresponding prompt will be skipped in the setup wizard.
-k	--key	<filename>	Specifies a private key file to be used for HTTPS. This must be specified in conjunction with the -c switch. If an SSL/TLS certificate and private key are specified as command line parameters, the corresponding prompt will be skipped in the setup wizard.
-l	--license	<filename>	Specifies a license file name containing the license details. Specifying this as a command line parameter will cause the corresponding prompt to be skipped in the setup wizard.
-o	--output	<filename>	Specifies the name of the JSON configuration file to which the configuration will be saved. Specifying this as a command line parameter will cause the corresponding prompt at the end of the setup wizard to be skipped.

trackhound-solo

trackhound-solo is a standalone version of PTP Track Hound v2 that operates without a separate capture service running in the background. It accepts the following parameters:

-h	--help		Displays usage information for trackhound-solo.
-b	--binary	<filename>	Normally trackhound-solo will use the trackhound-service executable located in the same directory as trackhound-solo. If for any reason you wish to use another version of trackhound-service, the path to it can be specified here.
-c	--config	<filename>	Specifies a JSON-formatted configuration file for Solo Mode.
-s	--store	<filename>	Specifies a custom JSON store file in which capture and analysis data will be stored.

trackhound-certgen

`trackhound-certgen` is used to generate a self-signed SSL/TLS certificate for HTTPS access to the Web Interface and REST API. It accepts the following parameters:

`-h`	`--help`		Displays usage information for `trackhound-certgen`.
`-n`	`--name`	`<name>`	Common name (hostname) to be protected by the certificate. This name will also be added as the first entry of the SAN list.
`-o`	`--organization`	`<name>`	Name of the organization
`-u`	`--unit`	`<name>`	Name of the organizational unit or division (e.g., "*Software Development*")
`-e`	`--email`	`<email>`	Contact email address of person responsible for certificate
`-c`	`--country`	`<code>`	Country name (two-letter code, e.g., "*DE*")
`-s`	`--state`	`<name>`	State or province name
`-l`	`--locality`	`<name>`	Locality (i.e., where your organization is based)
`-d`	`--domain`	`<domain>`	Additional domain name to be added to the SAN list. Multiple additional domain names must each individually be preceded by `-d` switches.
`-a`	`--all`		Add all currently assigned IP addresses to the SAN list.
`-i`	`--ip`	`<ip-address>`	Additional IP address to be added to the SAN list. Multiple additional IP addresses must each individually be preceded by `-i` switches.
`-v`	`--validity`	`<days>`	Validity of the certificate in number of days.
`-r`	`--request`		Creates a certificate signing request (CSR) for your CA.
`-f`	`--filename`	`<filename>`	Name of the output certificate file. This must be used in conjunction with the `-k` switch.
`-k`	`--key`	`<filename>`	Name of the output key file. This must be used in conjunction with the `-f` switch.

14.2 Types of Analysis in PTP Track Hound v2

PTP Track Hound employs four different types of analysis in order to provide three different types of information collected from different perspectives. In addition to the basic analysis, PTP Track Hound also supports **Management Messages**, **Capture Time Offsets**, and **NetSync Monitor**.

Basic Analysis

At the most basic level, PTP Track Hound analyzes PTP nodes using the data provided in the conventional *Announce*, *Sync*, *Follow Up*, *Delay Request*, and *Delay Response* messages. This allows PTP Track Hound to determine the following information for each PTP node:

- the current time,
- the node's port state,
- if in timeReceiver mode, the node's Grandmaster,
- the number of hops between the timeReceiver and Grandmaster
- the primary time reference of the PTP node (e.g., GPS)
- whether a time or frequency reference is traceable back to a primary reference source

Management Messages

When Management Messages are enabled, PTP Track Hound will periodically transmit a Management Message as frequently as defined in the settings—the default setting is every 60 seconds. This Management Message prompts the receiving PTP nodes to transmit the requested dataset, which in addition to the standard Best TimeTransmitter Clock Algorithm data also contains information about a timeReceiver node's current offset to its timeTransmitter, the mean path delay between timeTransmitter and timeReceiver, information about the manufacturer, firmware version, and more.

In the context of PTP Track Hound, the data generated by PTP Management Messages is therefore referred to as **reported data**, as it relies on data provided by the PTP nodes themselves to assess their performance.

Of the three measurement methods supported by PTP Track Hound, management messages are the only one that is a native feature of the IEEE1588 PTP standard, specifically through the use of TLVs, or type-length-value extensions, that allow additional data to be conveyed as part of standard and also non-standard extensions. As such, it has the benefit of being the most widely supported of the three methods.

However, there are two key drawbacks to using Management Messages in isolation. Firstly, networks may disable the distribution of PTP Management Messages as a potential security risk—after all, Management Messages can be used to alter a clock's behavior. Secondly, there is a certain amount of (potentially misplaced) trust involved in the use of the data provided by a clock. If a clock improperly generates its datasets in order to encourage or force a certain outcome, this can in turn distort the statistics generated by PTP Track Hound.

Management Messages can be a valuable tool if your network's clocks provide correct and trustworthy data, but are best interpreted in the context of the other two measurement methods described below.

NetSync Monitor

When NetSync Monitor support is enabled, PTP Track Hound will periodically send a custom *Delay Request* message, the frequency of which is defined in the settings. This message contains special TLV data (see *Management Messages* above for more information) that devices with NetSync Monitor support can interpret. A device receiving and processing this *Delay Request* message will react by first transmitting a custom *Delay Response* message (which itself contains a special TLV), followed by a Sync message and—if the clock is operating in two-step mode—a *Follow Up* message.

In doing so, the receiving clock has provided not only its own path delay, but also its current time. PTP Track Hound uses this data to measure offsets between the PTP node itself and the PTP Track Hound instance's local clock. It can also compare the data against that of the corresponding timeTransmitter in order to understand more about the relationship between timeTransmitter and timeReceiver.

In the context of PTP Track Hound, the data generated by NetSync Monitor implementations is referred to as **measured data**, as it allows PTP Track Hound to measure performance data on a first-hand basis.

As such, NetSync Monitor provides the most reliable data for assessing PTP node performance. **Most** (but not all) information provided by Management Messages is also available through NetSync Monitor, and may under certain circumstances render Management Messages superfluous.

However, like Management Messages, NetSync Monitor has a couple of drawbacks.

Firstly, support for NetSync Monitor is not widespread at time of writing (although if you only use Meinberg or Oregano Systems clocks in your network, you can be assured that every clock in your network supports NetSync Monitor).

Secondly, NetSync Monitor requires the clock of the PTP Track Hound instance to itself be synchronized to a separate time server (but ideally the same time reference, e.g., GPS) for measurements to be meaningful. Without this synchronization, there is no point of reference for changes in the time provided by the clock.

Capture Time Offsets

Capture Time Offsets involve timestamping every incoming PTP message with the current time of the PTP Track Hound instance's local clock. The absolute offset is actually not particularly relevant in this case—the more meaningful information is provided by how these offsets evolve over time.

If the Capture Time Offset of all PTP messages from a certain PTP node remains relatively constant over a prolonged period, this is indicative of ideal network conditions with minimal jitter and a stable time reference. Conversely, if the Capture Time Offset delta value is highly volatile, this may indicate problems with network congestion or poor PTP traffic prioritization (for example, through a non-PTP-aware switch). Sudden changes in Capture Time Offset delta values after a period of stability may also be indicative of drift or jumps resulting from adverse changes in a timeTransmitter.

Like NetSync Monitor, Capture Time Offsets also require the device on which PTP Track Hound is running to be synchronized—the device's local clock needs to be synchronized to a separate time server (but ideally the same time reference, e.g., GPS). However, in contrast to NetSync Monitor, Capture Time Offset functionality benefits from not relying on any specific functionality of the PTP nodes, such that it can be used with any PTP clock.

Summary of Benefits and Drawbacks

Ideally, all three measurement methods should be used for the most complete analysis. However, it is important to understand how the data from each method relates to one another.

The chief benefit of Management Messages is their ubiquity. Because they are specified by the IEEE1588-2008 standard, almost all PTP clocks on the market support them. They are the only way to acquire certain data directly from the clocks, for example device metadata or which timescale a clock is using (PTP or arbitrary). It is also the only one of the three methods that does not require the local clock of the PTP Track Hound host device to be synchronized.

However, there is a certain element of 'trust' involved in management messages that may or may not be warranted with certain PTP devices. NetSync Monitor overcomes this by allowing some data provided by Management Messages to be verified first-hand, with clock offset and path delay being measured directly by PTP Track Hound. Of course, data outside of the Current Dataset (such as the configured Clock Class, timeReceiver-only states, steps removed from timeTransmitter) remain reported.

If all of the nodes monitored by PTP Track Hound fully support NetSync Monitor and respond appropriately to NetSync Monitor requests, Management Messages may not even be necessary. Management Messages could be temporarily activated to acquire the device metadata and then disabled.