



The Synchronization Experts.



MANUAL

PTP Track Hound

Version 2.0.4

December 22, 2022

Meinberg Funkuhren GmbH & Co. KG

Table of Contents

1	Imprint	1
2	User Guide Revision History	2
3	Copyright and Liability Exclusion	3
4	Introduction to PTP Track Hound v2	4
4.1	Terminology of Navigation Elements of the PTP Track Hound Version 2 Web Interface	5
4.2	Formatting and Structural Principles of this Manual	6
5	Getting Started	7
5.1	System Requirements	7
5.2	License Types	9
5.3	Installation	11
5.3.1	Installation under Windows	11
5.3.2	Installation under Debian-Based Linux Distributions	15
5.3.3	Installation under Red Hat Enterprise Linux and Other RPM-Based Distributions	17
5.3.4	Installation on Raspberry Pi	19
5.3.5	Installation under macOS	21
5.4	Initial Configuration Using Setup Wizard	23
5.5	Launching and Logging In	29
5.6	Activating or Swapping a License	31
5.7	Changes from Version 1	33
6	General Operation	35
6.1	Basic Configuration and Operation Principles	35
6.2	Live Capture of PTP Traffic in Network	36
6.3	Importing pcap Capture Files	37
6.4	Exporting pcap Capture Files	38
6.5	Changing Password	39
6.6	Logging Out	39
7	Dashboard	40
8	Traffic	48
9	Scopes	55
10	Devices	59
11	Settings	61
11.1	Packet Capture	62
11.2	Remote Capture	66
11.3	Notifiers and Alarms	67
11.4	Web Server (Web Interface Only)	71
11.5	Files (Solo Mode Only)	73
11.6	Terminology	74
11.7	User Management	75
11.8	Memory Usage	75
11.9	Logging	76
12	REST API Reference	77
12.1	/api/cache	79
12.1.1	/api/cache/devices/[_id]	79
12.1.2	/api/cache/segments/[_id]	80

12.1.3	/api/cache/vendors/[_id]	80
12.2	/api/captures/[_id]	81
12.2.1	/api/captures/management	82
12.3	/api/config	83
12.3.1	/api/config/api	86
12.3.2	/api/config/capture	87
12.3.3	/api/config/evaluation	89
12.3.4	/api/config/event	89
12.3.5	/api/config/license	91
12.3.6	/api/config/logging	91
12.3.7	/api/config/memory	91
12.3.8	/api/config/remote	92
12.3.9	/api/config/terminology	92
12.3.10	/api/config/users/[_id]	92
12.4	/api/current	93
12.4.1	/api/current/devices/[_id]	98
12.4.2	/api/current/instances/[_id]	99
12.4.3	/api/current/packets	103
12.4.4	/api/current/scopes/[_id]	105
12.4.5	/api/current/segments/[_id]	106
12.5	/api/events/[_id]	108
12.5.1	/api/events/[_id]/time	108
12.6	/api/info	109
12.6.1	/api/info/license	109
12.7	/api/memory	110
12.7.1	/api/memory/used	110
12.8	/api/network	111
12.8.1	/api/network/interfaces/[_id]	111
12.9	/api/files	112
12.9.1	/api/files/capture	112
12.9.2	/api/files/img	112
13	Appendix	113
13.1	Operation from Command Line Interface	113

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 21, 2022

2 User Guide Revision History

Version	Date	Revision Notes
1.0	10/19/2022	Initial version
1.1	11/01/2022	<ul style="list-style-type: none">- 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	<ul style="list-style-type: none">- 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

3 Copyright and Liability Exclusion

Except where otherwise stated, the contents of this document, including text and images of all types and translations thereof, are the intellectual property and copyright of Meinberg Funkuhren GmbH & Co. KG ("Meinberg" in the following) and are subject to German copyright law. All reproduction, dissemination, modification, or exploitation is prohibited unless express consent to this effect is provided in writing by Meinberg. The provisions of copyright law apply accordingly.

Any third-party content in this document has been included in accordance with the rights and with the consent of its copyright owners.

A non-exclusive license is granted to redistribute this document (for example, on a website offering free-of-charge access to an archive of product manuals), provided that the document is only distributed in its entirety, that it is not modified in any way, that no fee is demanded for access to it, and that this notice is left in its complete and unchanged form.

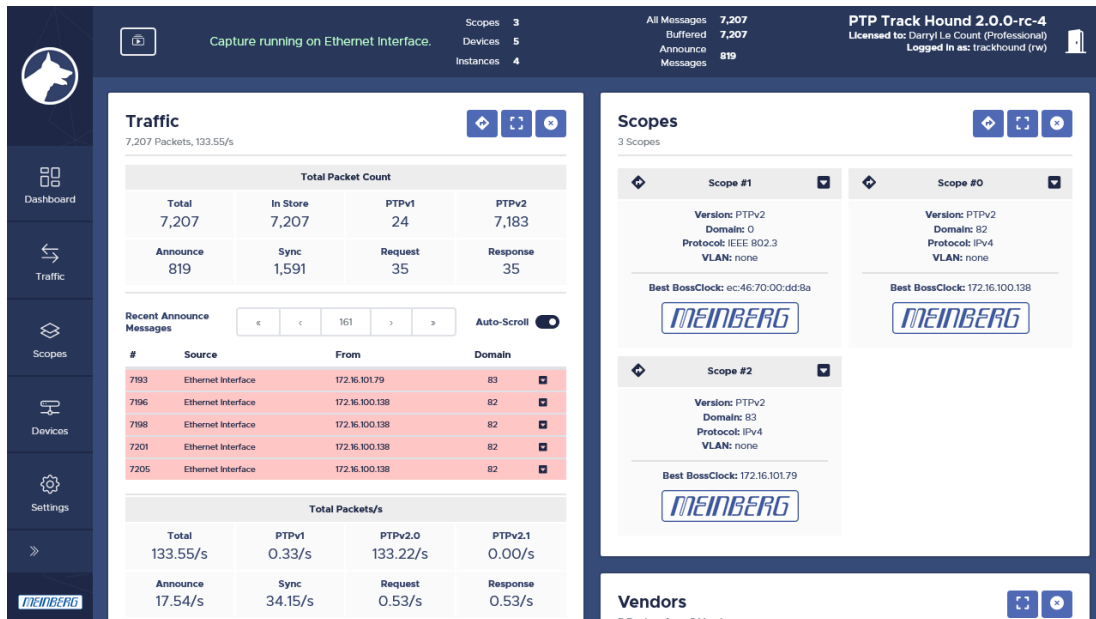
At the time of writing of this document, reasonable effort was made to carefully review links to third-party websites to ensure that they were compliant with the laws of the Federal Republic of Germany and relevant to the subject matter of the document. Meinberg accepts no liability for the content of websites not created or maintained by Meinberg, and does not warrant that the content of such external websites is suitable or correct for any given purpose.

While Meinberg makes every effort to ensure that this document is complete, suitable for purpose, and free of material errors or omissions, and periodically reviews its library of manuals to reflect developments and changing standards, Meinberg does not warrant that this specific document is up-to-date, comprehensive, or free of errors. Updated manuals are provided at www.meinbergglobal.com.

You may also write to techsupport@meinberg.de to request an updated version at any time or provide feedback on errors or suggested improvements, which we are grateful to receive.

Meinberg reserves the right to make changes of any type to this document at any time as is necessary for the purpose of improving its products and services and ensuring compliance with applicable standards, laws & regulations.

4 Introduction to PTP Track Hound v2



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 three 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 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**.

Window refers to any layout element denoted by a white background and rounded corners containing information or options. This is not to be confused with the **browser window** as a whole, which is referred to as such where appropriate.

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](#)), and 11 ([Settings](#)).

Advanced information for application developers and introductory information for non-technical users is provided in the [Appendix](#).

Formatting

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

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

Filenames, possible values, and listed options for a configuration or status field are conventionally listed in *italics*. 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. 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, it is also possible to reduce the memory usage limit to reduce the RAM requirements.

Windows-Based PCs

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 with the desktop environment and 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 tested with the following web browsers:

Mozilla Firefox 105, Microsoft Edge 105, Google Chrome 105, Apple Safari 14, Chromium 105

5.2 License Types

PTP Track Hound v2 is provided with three different licensing models to suit a variety of needs.



Important!

When you purchase a 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.

Free License

The Free License is free of charge (subject to registration with Meinberg Funkuhren) 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

Basic License

The Basic License 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.

Professional License

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

- Transmission of PTP Management messages to request PTP datasets from clocks in the network to obtain data that cannot be conventionally obtained from passive monitoring of ordinary network traffic
- SNMP traps
- E-mail notifications of 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_2.0.4_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, 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 to the 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 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 *trackhound-certgen* tool; refer to Chapter 13.1, "[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 13.1, "[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**.

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. 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 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 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.

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 appear 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.

5.3.2 Installation under 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 Linux Mint 21, Linux Mint 20, 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_2.0.4_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_2.0.4_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_2.0.4_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.

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**" for more information.

6. You can now launch the PTP Track Hound v2 service using:

```
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.

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

```
dnf repoquery --requires /path/to/package/ptptrackhound-2.0.4-1.x86_64.rpm
```

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*.

2. Install the RPM package with the following command:

```
sudo dnf install /path/to/package/ptptrackhound-2.0.4-1.x86_64.rpm
```

Important!



If you are updating an existing PTP Track Hound v2.0.0 installation to v2.0.1, please note that there is a bug in the v2.0.0 RPM package that will prevent v2.0.1 from installing properly. This can be fixed by entering the following after the RPM package is installed as above:

```
sudo dnf reinstall ./ptptrackhound-2.0.4-1.x86_64.rpm
```

This bug is fixed as of the v2.0.1 RPM package.

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 ptpttrackhound2 -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 "*ptpttrackhound2*" in your *HOME* directory.

The **Common Name** (*ptpttrackhound2*) 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.

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**" for more information.
6. You can now launch the PTP Track Hound v2 service using:

```
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_2.0.4_armhf.deb
```

or

```
dpkg --info /path/to/package/ptptrackhound_2.0.4_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_2.0.4_armhf.deb
```

or

```
sudo apt install /path/to/package/ptptrackhound_2.0.4_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_2.0.4_arm64.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 ptpttrackhound2 -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 "*ptpttrackhound2*" in your *HOME* directory.

The **Common Name** (*ptpttrackhound2*) 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.

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](#)" for more information.

6. You can now launch the PTP Track Hound v2 service using:

```
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 received a link by email.
2. Install the package with the following command:

```
sudo ./path/to/package/ptptrackhound_2.0.4_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_2.0.4_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.

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**" 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 11, "**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 sub-domain, and to a given network protocol (any, IPv4, IPv6, IEEE802.3), and the IPv6 multicast scope for management messages. Refer to Chapter 11, "**Settings**" for more information. This feature requires a **Professional 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.

For more information on configuring network interfaces, refer to Chapter 11.1, "**Packet Capture**".

2. `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 11.1, "Packet Capture".

3. `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 **Basic 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 11.2, "Remote Capture".

4. `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.

5. `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 Master/Leader to Slave/Follower), when the local quality changes, when the Grandmaster quality changes, when the grandmaster is changed, and when a custom alarm is triggered or cleared.

For more information on configuring notifications in PTP Track Hound v2, refer to Chapter 11.3, "[Notifiers and Alarms](#)".

6. `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 notifications channels. This feature requires a **Professional License**.

Custom alarms are defined using REST API resource routes.

Please refer to Chapter 12, "[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 11.3, "[Notifiers and Alarms](#)".

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

If answered with *y*, PTP port states will use the terminology **timeTransmitter**, **Pre-timeTransmitter**, and **timeReceiver** instead of the standard IEEE 1588 terminology **Master**, **Pre-Master**, and **Slave**.

If you choose not to use the specified alternative terminology, you will be asked if you wish to specify your own terminology for these PTP port states. If you do not, the standard IEEE 1588 terminology will be used.

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

Please note that this does **not** affect the designations of the clocks themselves, which will continue to be declared using standard IEEE 1588 terminology.

For more information on configuring terminology in PTP Track Hound v2, refer to Chapter 11.6, "[Terminology](#)".

8. `Maximum memory (RAM) usage in MB (1 - 2048, 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.

9. `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 11.8, "Memory Usage".

10. `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.

11. `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.

12. `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.

13. `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.

14. `Log to standard streams (stdout/stderr)?`

If answered with 'y', log output with the specified maximum severity will be directed to the standard output stream of the *trackhound-service* process. This means that if the service is launched from a console, all log output with the specified severity will be output through that console. While it is possible to redirect the output to a file, a more elegant solution is to enable the "log to file" option (see Step 16 below).

You will be prompted to specify the maximum severity for *stdout/stderr* output.

15. Log to file (y/n)?

If answered with 'y', log output with the specified maximum severity will be written to the specified file.



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.

You will be prompted to specify the maximum severity for file output.

16. Log to local syslog (y/n)?

If answered with 'y', log output with the specified maximum severity will be directed under Linux or macOS to the local *syslog* instance. Under Windows, it will be directed to the Events Log (under the *cygwin* process).

17. 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, 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.

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

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

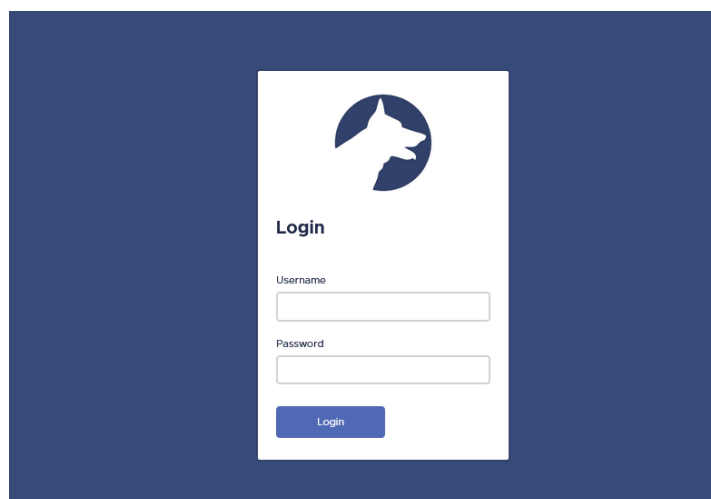


Figure 5.1: PTP Track Hound v2 Login Page

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 **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`

Solo Mode

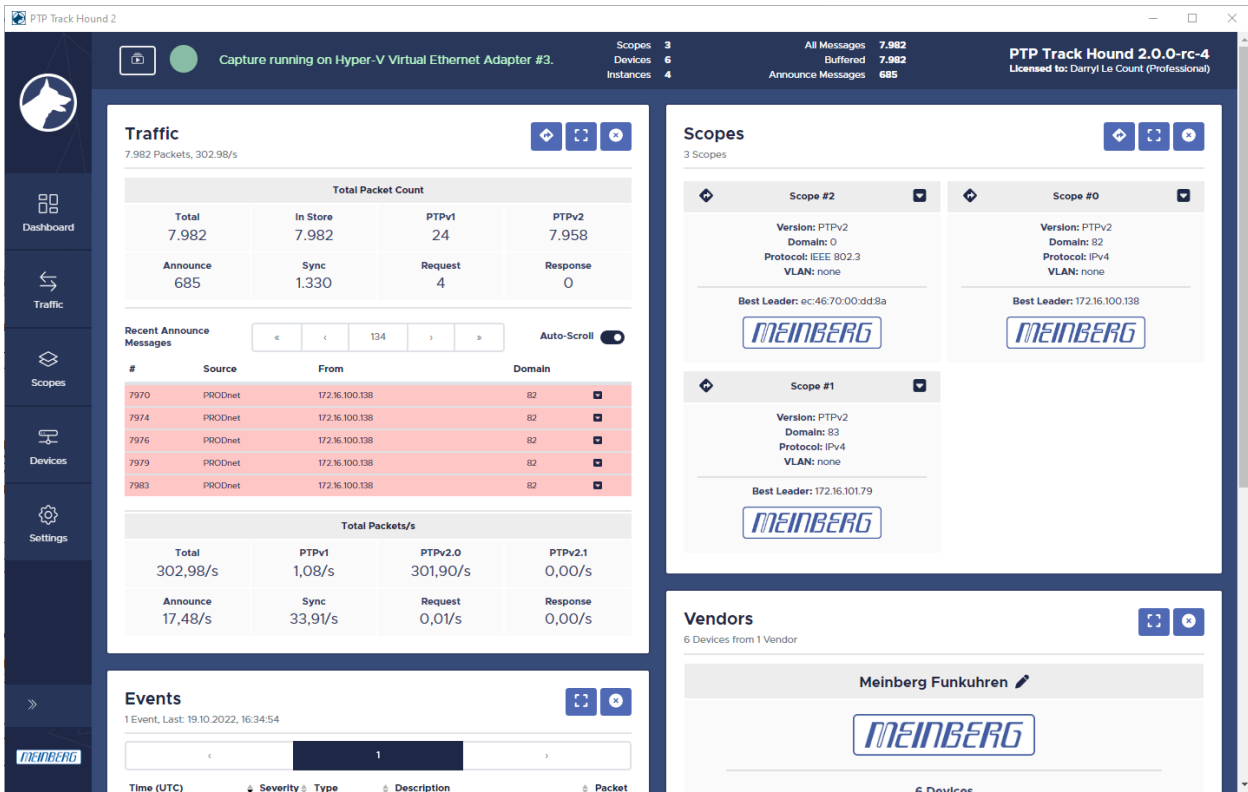


Figure 5.2: PTP Track Hound v2 Solo Mode

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

If you have purchased a PTP Track Hound v2 Basic or Professional License or received a trial license but did not activate it during installation for any reason, or if you wish to amend one of your licenses (for example to move a Professional License to another device, or to replace a trial license with a full license), you may (re-)register your license through the Web Interface.

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".

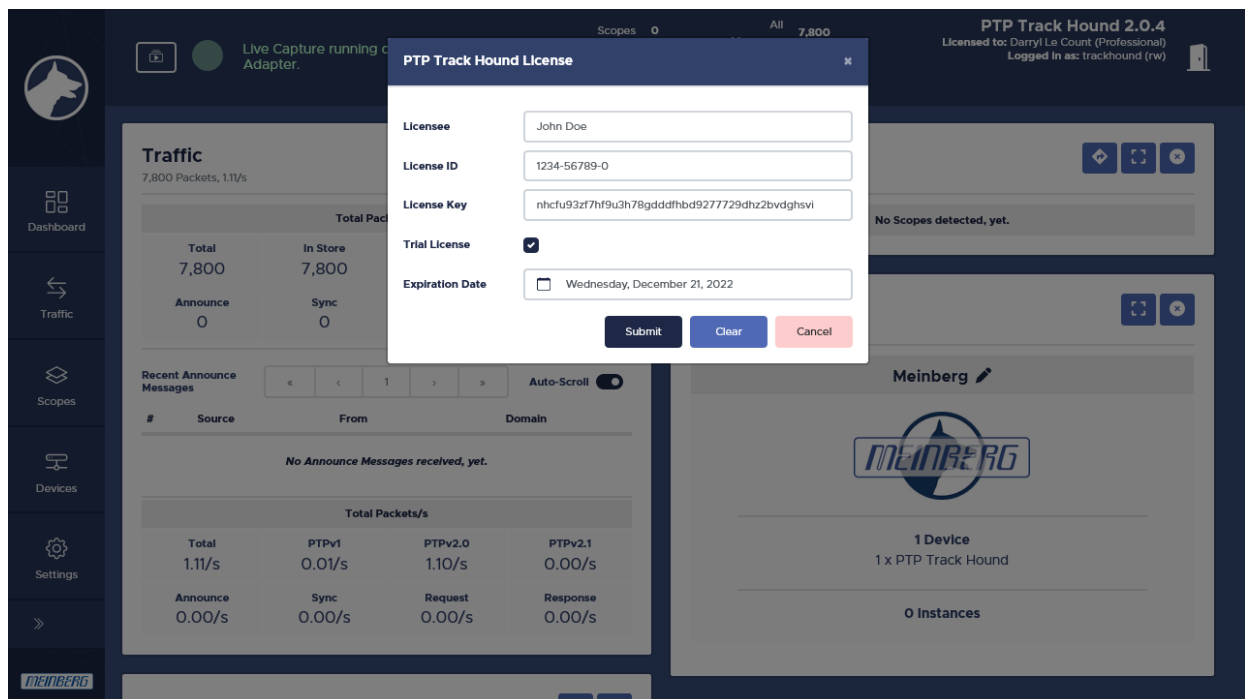


Figure 5.3: Registering a License

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 provided to you by Meinberg.



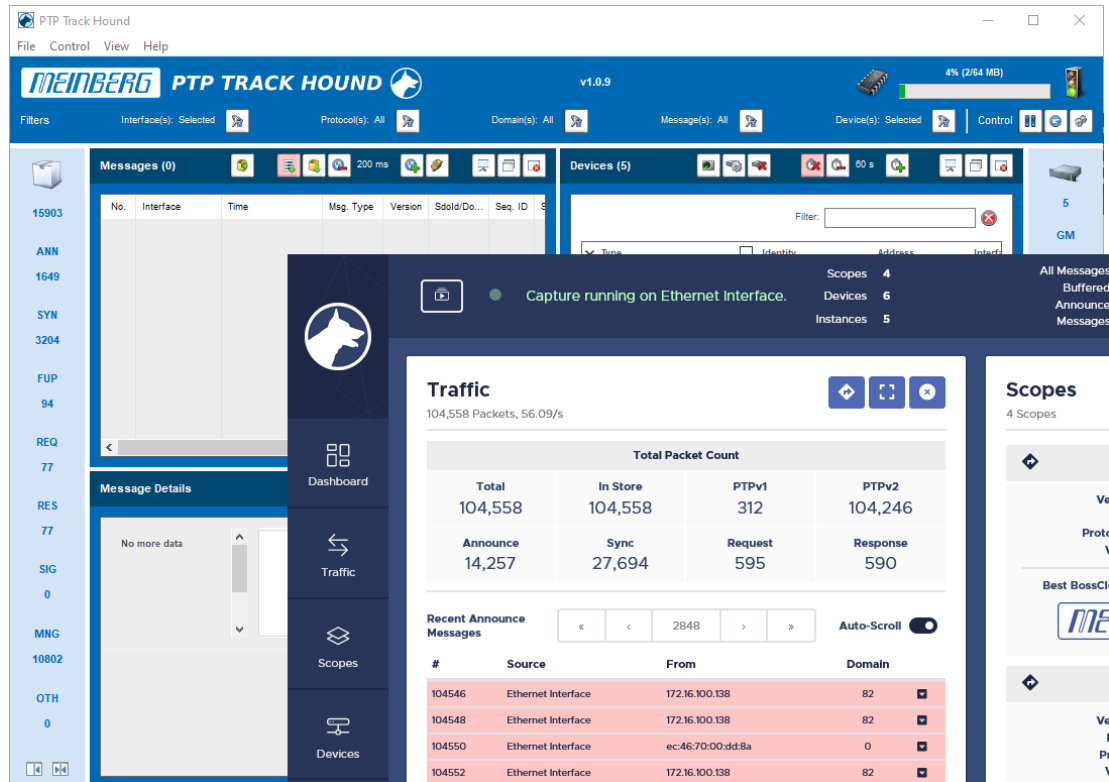
Important!

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



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 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 Slave Clocks as subordinate to it, Version 2 treats PTP master/slave 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 Dashboard.

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.

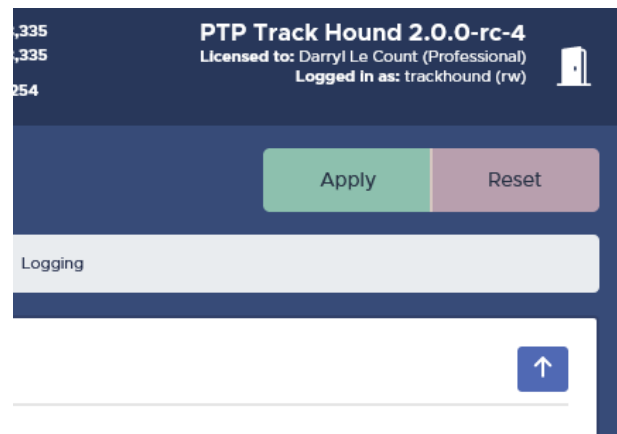
Finally, because this new Version 2 of PTP Track Hound is focused on the provision of enterprise-grade remote monitoring features, it also now also features a multitude of logging and notification features, including user-definable alarm conditions and comprehensive SNMP trap, syslog, and email notification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

6 General Operation

6.1 Basic Configuration and Operation Principles

Most 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"**.



Important!

Certain configuration panels, namely the **"Terminology"** and **"User Management"** panels, 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 server.

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

Each window 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 window.



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



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



The "Close" button hides the window.



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



The "Reset View" button at the bottom right of the page restores the default window arrangement by displaying any windows that were previously closed. This button is only displayed if there are windows hidden.



The "Closed Windows" button at the bottom right of the page allows you to select a hidden window to be restored to the page. This button is only displayed if there are windows hidden.

6.2 Live Capture of PTP Traffic in Network

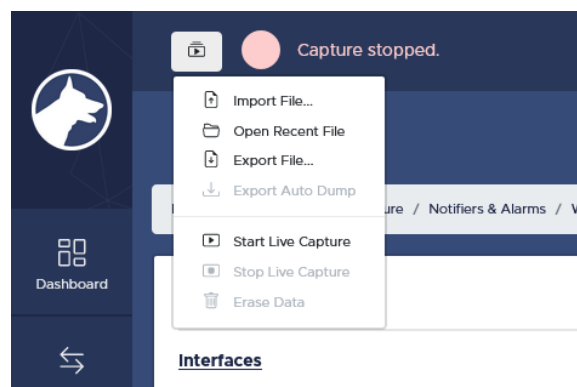
The core function of PTP Track Hound v2 lies in its ability to capture live PTP traffic from any or all of the network interfaces of the devices on which PTP Track Hound v2 is running and to dynamically analyze it.

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!

3. If the file is successfully imported, you will see a prompt in the Header Bar that it has been opened.
4. The captured data is now ready for review using the **Traffic**, **Scopes**, and **Devices** sections.

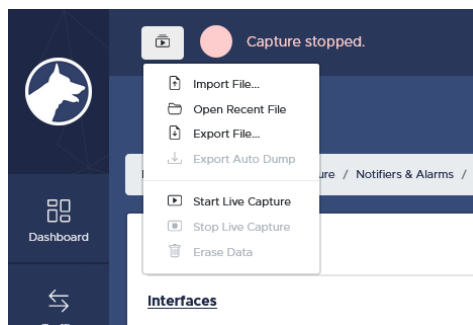
6.4 Exporting pcap Capture Files

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

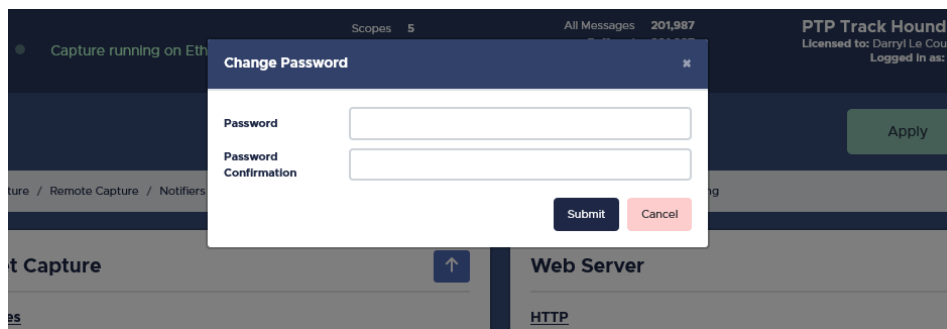


Figure 6.1: Changing the Password

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 11, "**Settings**" for further information.

6.6 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.

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

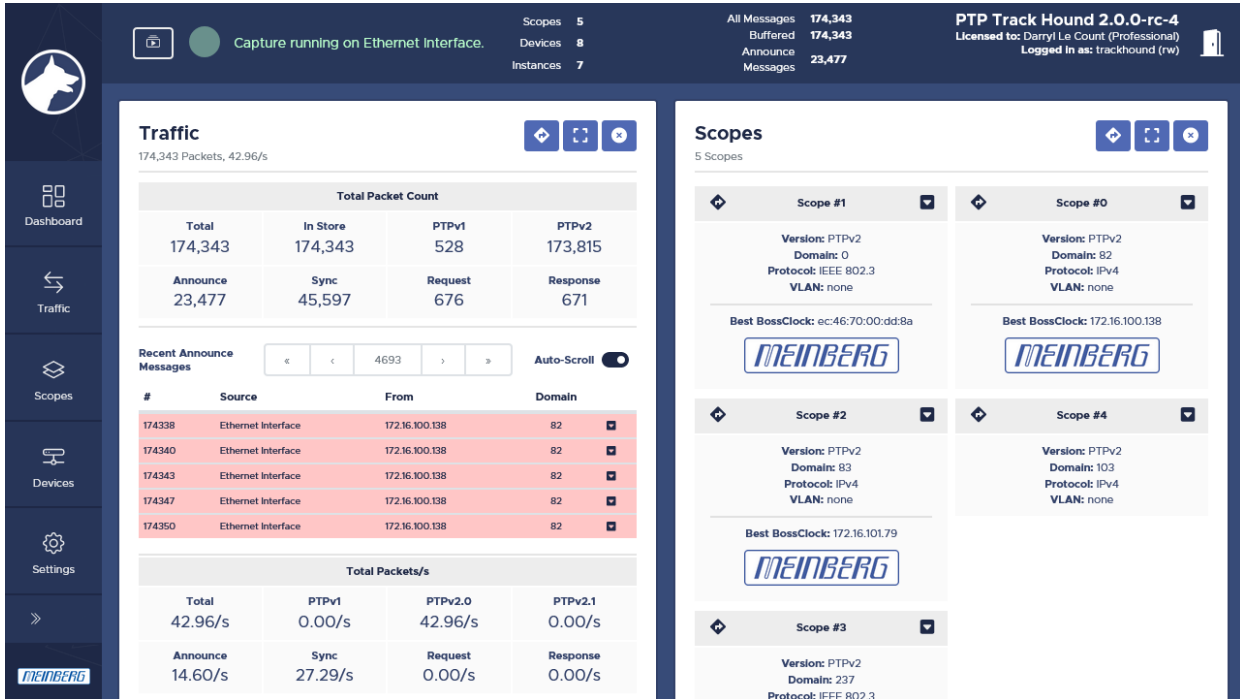


Figure 7.1: Dashboard

The **Dashboard** provides a general overview of the captured traffic and the results of the analyses performed on this traffic. It comprises five windows: **Traffic**, **Events**, **Memory**, **Scopes**, and **Vendors**.

Traffic

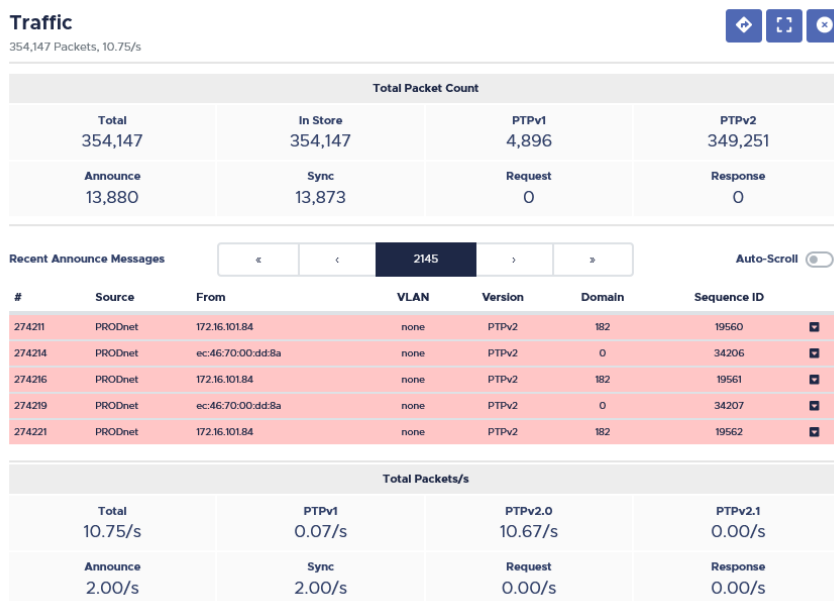


Figure 7.2: Traffic Window

The **Traffic** window displays the most recent PTP *Announce* messages captured by the capture service or, if a capture file is being analyzed, the *Announce* messages in that capture file. It also provides a number of relevant packet statistics.

The **Total Packet Count** section lists the total number of PTP messages received by the capture service and currently held in memory, and a breakdown of these messages by PTP version and message type.

The **Recent Announce Messages** section lists the most recent PTP *Announce* messages received by the capture service or, if a capture file is being analyzed, the *Announce* messages contained in that file. The list is regularly refreshed to show the latest *Announce* 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 *Announce* message list.

#	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.
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.



Figure 7.3: Breakdown of a PTP Message

If the **Auto-Scroll** toggle switch is disabled, a breakdown of any given *Announce* message can also be displayed by clicking on the **Expand** button of the *Announce* message line. This 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.
- 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.
- Type** The type of PTP message, which in this case will always be *Announce*.
- 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).

Ethernet II	IPv4	UDP	PTPv2	Announce
Destination Address		0100:5e:00:01:81		
Source Address		ec:46:70:00:9c:f2		
Protocol		0x0800 (IPv4)		

Figure 7.4: Layer-by-Layer Breakdown

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.

Total Packets/s			
Total	PTPv1	PTPv2.0	PTPv2.1
42.96/s	0.00/s	42.96/s	0.00/s
Announce	Sync	Request	Response
14.60/s	27.29/s	0.00/s	0.00/s

Figure 7.5: Statistics: Total Packets per Second

The **Total Packets/s** section shows the regularity of PTP messages captured by the capture service per second, as well as a breakdown by PTP version and message type. If a capture file is being analyzed, these statistics will be derived from the timestamp data present in the capture file.



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



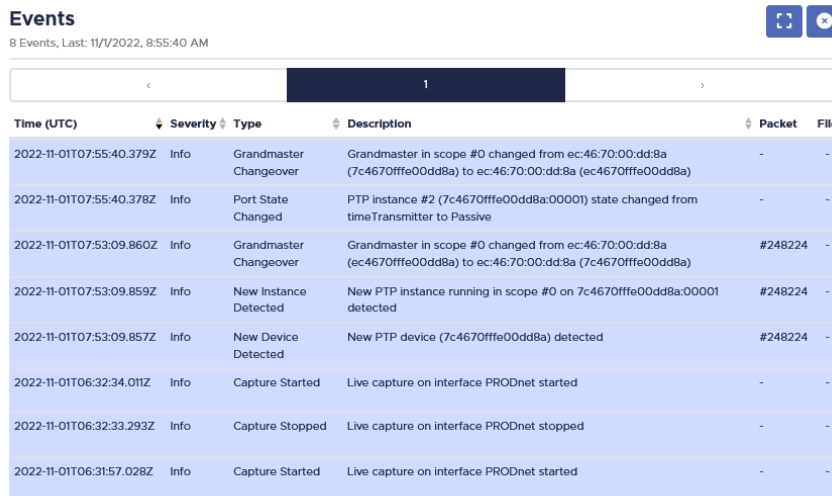
You may also bring up a window showing more detailed message information by clicking on the **Show Details** button, which will automatically bring you to the **"Traffic"** section (see Chapter 8).



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

For more information, refer to Chapter 8, **"Traffic"**.

Events



Time (UTC)	Severity	Type	Description	Packet	File
2022-11-01T07:55:40.379Z	Info	Grandmaster Changeover	Grandmaster in scope #0 changed from ec:46:70:00:dd:8a (7c4670ffe00dd8a) to ec:46:70:00:dd:8a (ec4670ffe00dd8a)	-	-
2022-11-01T07:55:40.378Z	Info	Port State Changed	PTP instance #2 (7c4670ffe00dd8a:00001) state changed from timeTransmitter to Passive	-	-
2022-11-01T07:53:09.860Z	Info	Grandmaster Changeover	Grandmaster in scope #0 changed from ec:46:70:00:dd:8a (ec4670ffe00dd8a) to ec:46:70:00:dd:8a (7c4670ffe00dd8a)	#248224	-
2022-11-01T07:53:09.859Z	Info	New Instance Detected	New PTP instance running in scope #0 on 7c4670ffe00dd8a:00001 detected	#248224	-
2022-11-01T07:53:09.857Z	Info	New Device Detected	New PTP device (7c4670ffe00dd8a) detected	#248224	-
2022-11-01T06:32:34.011Z	Info	Capture Started	Live capture on interface PRODnet started	-	-
2022-11-01T06:32:33.293Z	Info	Capture Stopped	Live capture on interface PRODnet stopped	-	-
2022-11-01T06:31:57.028Z	Info	Capture Started	Live capture on interface PRODnet started	-	-

Figure 7.6: Events Window

The **Events** window displays the most recent events detected in relation to PTP clocks and instances.

Examples of events include clocks detected, clocks no longer being found, changes to the Grandmaster, identification of PTP scopes, the capture service being started and stopped, or capture files being loaded. It relates only to the current capture file or capture session; whenever a live capture is (re)started, the events list will be cleared.

The **Time** column shows the time (according to the clock of the capturing instance) at which the event was logged.

The **Severity** column shows the severity of the event, which may be set by PTP Track Hound v2 itself or, in the case of custom alarms or notifiers, by the configuration of that alarm or notifier. This affects whether specific events are written to the various log outputs. Refer to Chapter 11.9, "**Logging**" for more information.

The **Type** column specifies which type of event this is.

The **Packet** column shows which PTP message the event specific relates to (if applicable); clicking on the packet number will open the **Traffic** section and navigate to the corresponding message in the data store.

If Dashcam Mode is enabled and a dashcam file is generated for a specific event, a link to that dashcam file will be provided under the column **File**.



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



To hide this window, click on the "**Close Window**" button. This button will then remain hidden from view on the Dashboard until your browser's cookies are cleared or the window 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.

Memory

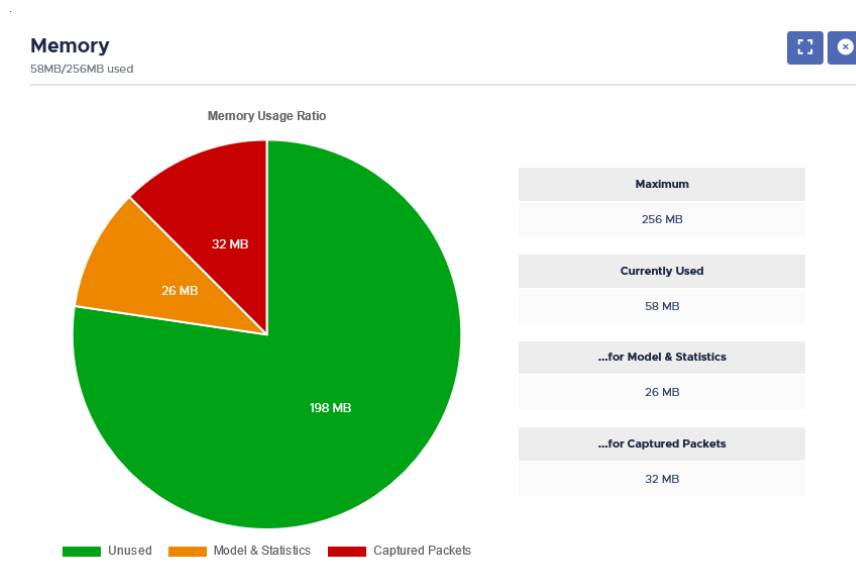


Figure 7.7: Memory Window

The **Memory** window provides a breakdown of how the configured memory buffer is being used.



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



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

Scopes

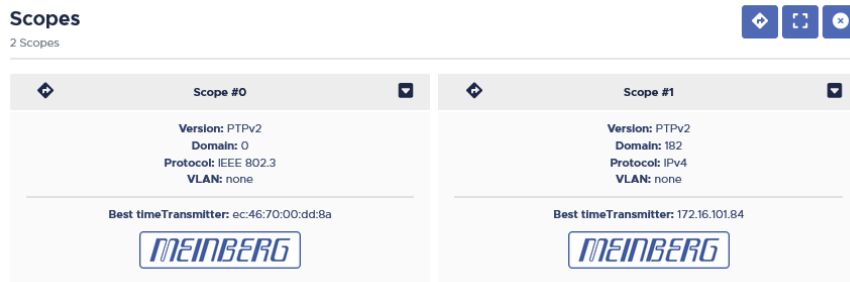


Figure 7.8: Scopes Window

A **scope** in PTP Track Hound v2 is defined as a collection of clocks that are capable of serving time to and receiving time from one another. The **Scopes** window shows the scopes that PTP Track Hound v2 has identified based on its analysis of the captured traffic.

All clocks within a scope must share a common network protocol, PTP 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.



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



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



You may also bring up a window showing more detailed scope information by clicking on the **Show Details** button, which will automatically bring you to the **"Scopes"** section (see Chapter 9). The details of any specific scope can also be opened directly in the Scopes page by clicking on the **Show Details** button of that scope to the *left* of the scope name.



Likewise, each scope panel can be expanded to include information on the common Grandmaster clock and the number of instances within that scope by clicking on the **Expand** button to the *right* of the scope name.

For more information, refer to Chapter 9, **"Scopes"**.

* Note that while scopes are defined strictly on the basis of PTP versions, clocks within an existing scope that change their PTP version from PTPv2.0 to PTPv2.1 will remain within their existing PTPv2 scope.

Vendors

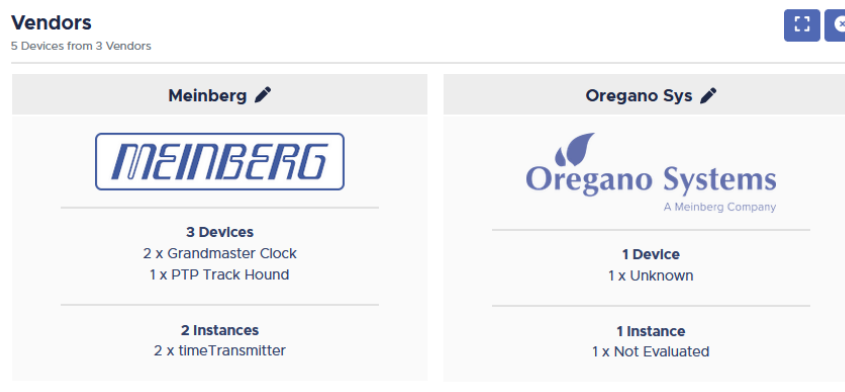


Figure 7.9: Vendors Window

The **Vendors** window shows an overview of the devices detected via the capture interface or in the capture file, grouped accordingly to the identified vendor. The vendor names will be displayed automatically based on the device's UID if the vendor is registered with IANA, or they can be manually entered into the device metadata via the Web Interface.



The metadata of any vendor displayed here can be edited by clicking on the **Edit** button.



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



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

For more information, refer to Chapter 10, "**Devices**".

8 Traffic

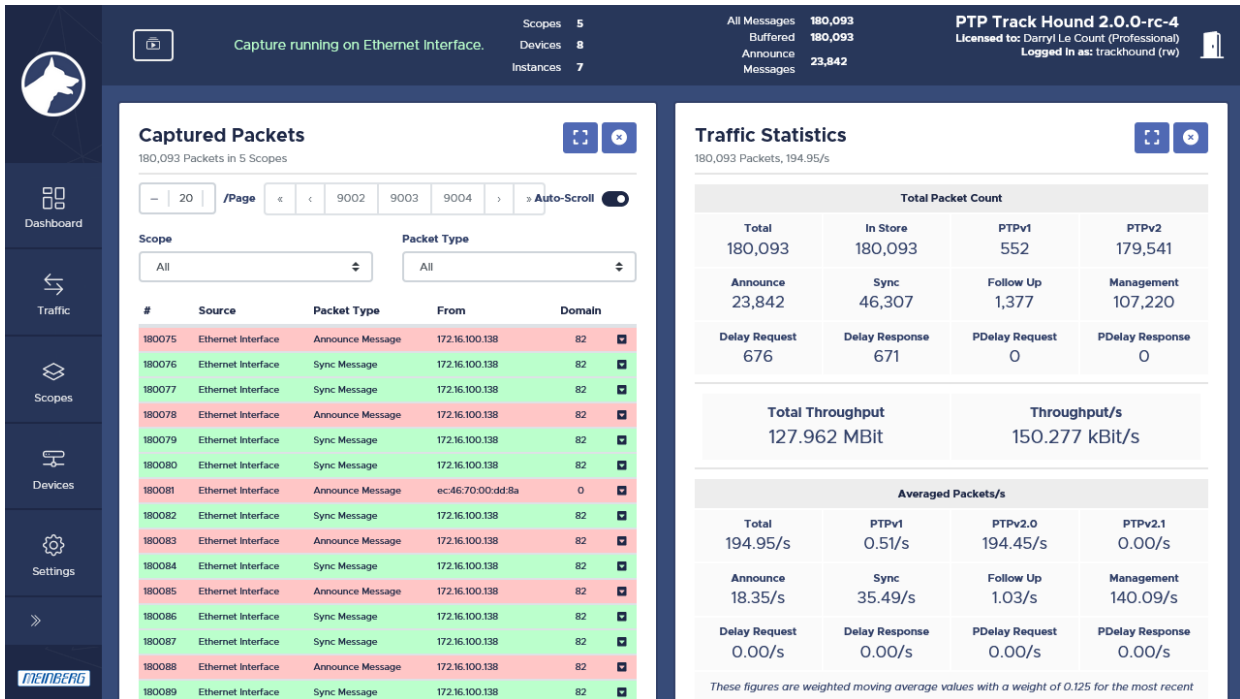


Figure 8.1: Traffic

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

Captured Packets

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



To hide this window, click on the **"Close Window"** button. This button will then remain hidden from view on the Dashboard until your browser's cookies are cleared or the window 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 window.

#	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.
Packet 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.

ID	#1592
Source	Ethernet Interface
Remote Packet	false
Capture Time	2022-11-01T10:30:15.546021
Processing Time	2022-11-01T10:30:15.546100
Type	Sync Message
Protocol	IEEE 802.3
VLAN	none
Version	PTPv2
Domain	0
Sequence ID	43129
From	ec:46:70:00:dd:8a
To	01:1b:19:00:00:00
Port Identity	ec4670ffe00dd8a:00001
Duplicate Packet	false

	Ethernet II		PTPv2		Sync	
Destination Address	01:1b:19:00:00:00					
Source Address	ec:46:70:00:dd:8a					
Protocol	0x88f7 (PTPv2 over Ethernet (IEEE 1588))					

0x0000	01	1b	19	00	00	00	ec	46	70	00	dd	8a	88	f7	00	02
0x0010	00	2c	00	00	02	00	00	00	00	00	00	00	00	00	00	00
0x0020	00	00	ec	46	70	ff	fe	00	dd	8a	00	01	a8	79	00	00
0x0030	00	00	63	60	f5	48	2a	88	83	47	00	00				

Figure 8.2: PTP Message Breakdown



If **Auto-Scroll** is disabled, a breakdown of any given message can also be displayed by clicking on the **Expand** button of the message line. This 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 (<i>true</i>) or locally on this PTP Track Hound v2 (<i>false</i>).
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.
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 (<i>IPv4, IPv6 or IEEE802.3</i>).
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 31.402 MBit		Throughput/s 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 8.3: Traffic Statistics

The **Traffic Statistics** window 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 window, click on the **"Close Window"** button. This button will then remain hidden from view on the Dashboard until your browser's cookies are cleared or the window 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 window. 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 window 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



Figure 8.4: Traffic History

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



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



To hide this window, click on the **"Close Window"** button. This button will then remain hidden from view on the Dashboard until your browser's cookies are cleared or the window 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 window 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 window. Click on + to add a graph, and - to remove the last graph in the window.

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 browser cookies; erasing your browser's cookies 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

PTP Track Hound 2.0.0-rc-4
Licensed to: Darryl Le Count (Professional)
Logged In as: trackhound (rw)

Capture running on Ethernet Interface.

Scopes: 5
Devices: 8
Instances: 7

All Messages: 190,199
Buffered: 190,199
Announce Messages: 25,656

Scopes

6 Instances in 5 Scopes

#	Version	Domain	Protocol	VLAN	Packets	Instances
1	PTPv2	0	IEEE 802.3	none	4,728	1

Current Grandmaster
ec:46:70:00:dd:8a (ec4670fffe00dd8a)

Priority 1	Clock Class	Clock Accuracy	Clock Variance	Priority 2
128	6	0x21	13563	128

State	Address	Vendor	Port Identity	Offset	Steps Removed
Grandmaster	ec:46:70:00:dd:8a	Meinberg Funkhuren	ec4670fffe00dd8a:1	-	0

MEINBERG

State: Grandmaster
Address: ec:46:70:00:dd:8a
Vendor: Meinberg Funkhuren
Port Identity: ec:4670fffe00dd8a:1
Delay Mechanism: EZE
Timestamp Mechanism: Two-Step

Clock Quality:
Priority 1: 128
Clock Class: 6
Clock Accuracy: within 100 ns (0x21)
Clock Variance: 13563
Priority 2: 128

Time Source: GPS
UTC Offset: 37 seconds
Leap Indicator 61: false
Leap Indicator 59: false
Timescale: PTP
Time Traceable: true
Emergency Traceable: true

Message Count:
Announce: 1480
Sync: 1480
Follow Up: 1480
(P)Delay Request: 0
(P)Delay Response: 0
Management: 144

Message Intervals:
Announce: 1/s
Sync: 1/s
(P)Delay Request: 1/s

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 one another. This means that all clocks within a scope must share a common network protocol, PTP 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 11, "**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

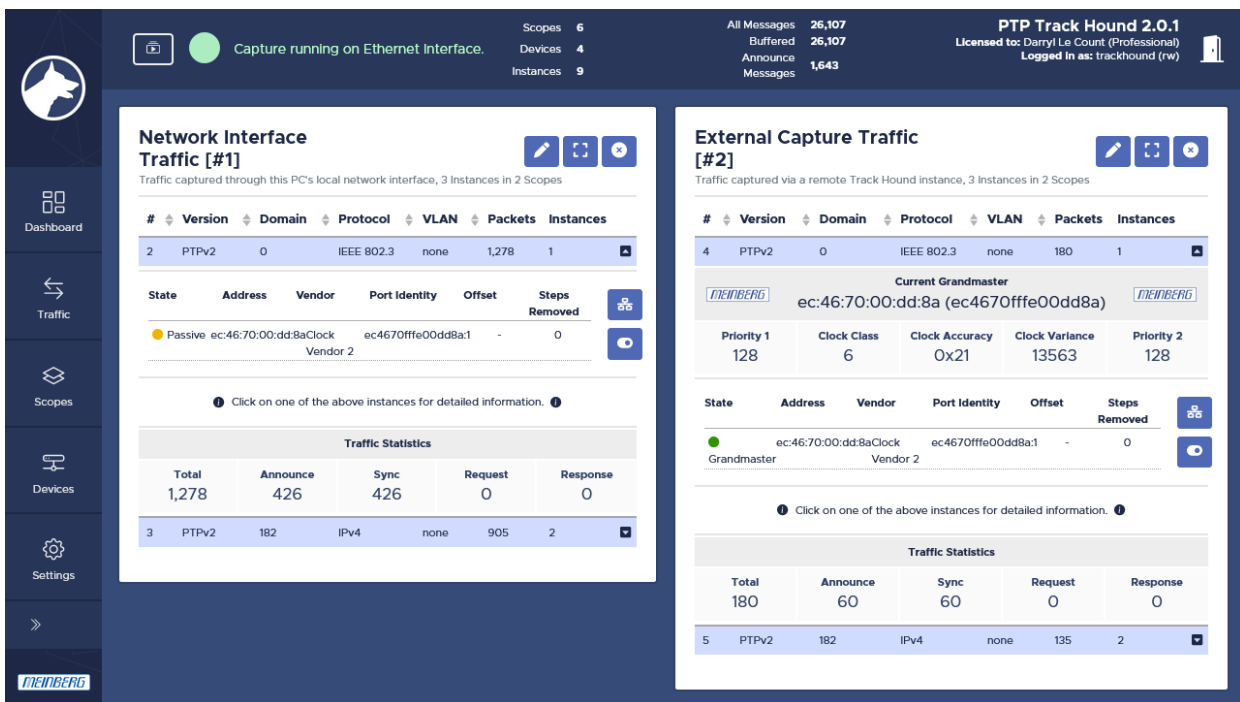


Figure 9.2: Scopes Broken Down into Segments

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 window 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 windows, 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 window.

This name and description will then be displayed at the top of each window 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 window, 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**".



If management messages are enabled, graphs can also be displayed for the measured mean path delay and the offset relative to its master clock by clicking on the "**Show Statistics**" button for any instance other than the Grandmaster clock while that instance is open.



The metadata for any given instance can be edited by clicking on the **Edit** icon next to the vendor 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.

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

The screenshot shows the PTP Track Hound 2.0.0-rc-4 interface. At the top right, it displays 'All Messages 196,165', 'Buffered 196,165', 'Announce Messages 26,091', and 'Licensed to: Darryl Le Count (Professional)'. The 'Devices' section is active, showing '8 Devices from 3 Vendors'. A filter dropdown is set to 'Entire Network: 8 Devices'. The network map on the left shows several devices connected by lines, with 'MEINBERG' devices highlighted. The list on the right shows the following data:

#	Type	Clock ID	Ports	Instances
2	Grandmaster Clock	ec4670ffe0aa249	1	1
1	Grandmaster Clock	ec4670ffe00dd8a	1	1
4	Grandmaster Clock	ec4670ffe00dde1	1	1
5	PTP Track Hound	b4b686290383712d	1	0
3	Unknown	ec4670ffe00bed4	1	1
8	Unknown	ec4670ffe00bdd1	1	1

Below the main list, there are two expanded sections:

Oxb4b686_Unknown				
#	Type	Clock ID	Ports	Instances
6	Unknown	b4b6862977838383	1	1

Ox88665a_Unknown				
#	Type	Clock ID	Ports	Instances
6	Unknown	b4b6862977838383	1	1

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 Settings

The screenshot displays the PTP Track Hound 2.0.0-rc-4 Settings interface. At the top, a status bar indicates 'Capture running on Ethernet Interface.' and provides system statistics: Scopes: 5, Devices: 8, Instances: 7, All Messages: 197,212, Buffered: 197,212, Announce Messages: 26,440. The user is logged in as 'trackhound (rw)'. The interface includes a sidebar with navigation options: Dashboard, Traffic, Scopes, Devices, and Settings. The main content area is split into two panels. The 'Packet Capture' panel shows a list of interfaces under the heading 'Interfaces'. The first interface is disabled, with ID '\Device\NPF_{EEA9721D-5F51-435C-9103-01EF2E12077B}', Friendly Name 'vEthernet (Default Switch)', Description 'Hyper-V Virtual Ethernet Adapter', and Addresses 'fe80::29e5:d925:fe97:163e/64, 172.19.208.1/16'. The second interface is selected, with ID '\Device\NPF_{A72DD2AD-406F-4D09-A1D3-206886779738}', Friendly Name 'PRODnet', Description 'Hyper-V Virtual Ethernet Adapter #3', and Addresses '172.16.100.109/16'. The 'Web Server' panel shows configuration for HTTP and HTTPS. HTTP is enabled with port 8080. HTTPS is also enabled with port 443, using certificates 'th2-cert.pem' and private keys 'th2-key.pem'. There is also a 'File Server' section partially visible.

Figure 11.1: 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 **Basic** or **Professional License**; this requirement is indicated where necessary.

11.1 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	<p>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.</p> <p>Segment IDs can also be used to isolate traffic from external PTP Track Hound v2 instances (see "Remote Connections" below).</p> <p>For more information on segmentation, refer to Chapter 9, "Scopes".</p>

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.

Send 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 **Professional 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 (<i>IPv4, IPv6, IEEE 802.3</i>). 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 (<i>PTPv1, PTPv2, PTPv2.1</i>). 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 (<i>_DFLT, _ALT1, _ALT2, _ALT3</i>).
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 (<i>0-255</i>).
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.

Remote Connections

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 11.2, "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 <i>AES-256</i> 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 different 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".

Miscellaneous

Initialization Phase	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 11.4, "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 11.4, "[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.

11.2 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 <i>32676</i> 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 11.1, " 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 <i>AES-256</i> 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.
---	---

11.3 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 11.1, " 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	<p>The security level used for SNMPv3: <i>noAuthNoPriv</i>, <i>authNoPriv</i>, or <i>authPriv</i>.</p> <p>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.</p>
Authentication Protocol	<p>Specifies the authentication protocol for SNMPv3 authentication: <i>MD5</i>, <i>SHA1</i>, <i>SHA224</i>, <i>SHA256</i>, <i>SHA384</i>, or <i>SHA512</i>.</p> <p>This requires a security level of <i>authNoPriv</i> or <i>authPriv</i>.</p>
Authentication Password	The authentication password of the SNMPv3 management solution.
Privacy Protocol	<p>Specifies the privacy protocol for SNMPv3 encryption: <i>DES</i>, <i>AES128</i>, <i>AES192</i>, or <i>AES256</i>.</p> <p>This requires a security level of <i>authPriv</i>.</p>

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 <i>514</i> .
Protocol	Specifies whether notifications should be sent to the syslog server via <i>UDP</i> or <i>TCP</i> .

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.
Custom Alarm Triggered	Triggers an event if a condition defined under Custom Alarms becomes <i>true</i> .
Custom Alarm Cleared	Triggers an event if a condition defined under Custom Alarms becomes <i>false</i> after it was previously <i>true</i>

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	<p>The condition is specified in three parts: the parameter from the REST API relative to the API root <i>/api</i>, a logical operator, and the operand to which the parameter will be compared. Chapter 12, "REST API Reference" provides a detailed list of all resources available via the REST API.</p> <p>For example, setting condition to <i>/api/current/instances/0/utcOffset</i>, operator to <i>is greater than</i>, and comparison value to <i>37</i> will trigger a custom alarm whenever the UTC offset of the instance 0 exceeds 37 seconds.</p>
Restrictions	<p>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.</p> <p>For example, the path <i>/api/current/instances/*/utcOffset</i> refers to the UTC offset values of every instance known to PTP Track Hound v2. Specifying a restriction of <i>/state</i> 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.</p>
Description	<p>The description consists of two components: a parameter name, and a JSON pointer to the object.</p> <p>For example, if a custom alarm refers to <i>/api/current/instances/*/localQuality/clockAccuracyValue</i>, a suitable description would be "Clock Accuracy" for description and <i>/address</i> as the object, which will output "Clock Accuracy of 192.168.20.20" to log files, for example.</p>

11.4 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 Wire-shark 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.

11.5 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 from the relevant directory.

11.6 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. Please note that this does not affect the designations of the clocks themselves, which will continue to be declared using standard IEEE 1588 terminology.

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.

11.7 User Management

This window 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!

11.8 Memory Usage

The **Memory Usage** window 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**".

By default, the **Maximum Memory Usage** 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.

11.9 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, custom alarms with severity <i>info</i>
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), custom alarms with severity <i>info</i> or <i>warning</i>
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 <i>info</i> , <i>warning</i> , or <i>error</i>
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 <i>info</i> , <i>warning</i> , <i>error</i> , or <i>debug</i>
trace	HTTP(S) session validations, periodical network inventory updates, service initialization status (completion time of initialization), all custom alarms

12 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 as necessary be substituted 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.

12.1 /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]*
-----> /clockID* (string)
-----> /id* (integer)
-----> /custom*
-----> /alias* (string)
-----> /firmwareRevision* (string)
-----> /hardwareRevision* (string)
-----> /softwareRevision* (string)
-----> /imagePath* (string)
-----> /location* (string)
-----> /modelName* (string)
-----> /vendor* (string)
-----> /segments*
-----> /[sequential index number]*
-----> /description* (string)
-----> /id* (integer)
-----> /name* (string)
-----> /vendors*
-----> /[sequential index number]* (integer)
-----> /id* (integer)
-----> /identifiers* (string array)
-----> /imagePath* (string)
-----> /name* (string)
```

* Directly referenceable endpoint

12.1.1 /api/cache/devices/[_id]

<code>/clockID</code>	<i>string</i>	The PTP clock ID of the device.
<code>/id</code>	<i>integer</i>	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.

12.1.1.1 /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 <code>/api/config/api/fileServer/directory</code> .
/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.

12.1.2 /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 .

12.1.3 /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 <code>/api/config/api/fileServer/directory</code> .
/name	string	The standard name used for the Vendor in the Web Interface and logging.

12.2 /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)
-----> /management
-----> /enabled (boolean)
-----> /flood (boolean)
-----> /interval (integer)
-----> /ipv6MulticastScope (string)
-----> /loopback (boolean)
-----> /protocol (string)
-----> /subdomain (string)
-----> /domain (integer)
-----> /version (string)
```

* Directly referenceable endpoint

/_index	<i>integer</i>	The sequential index number assigned in the resource level above.
/id	<i>integer</i>	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	<i>string</i>	The filename of an externally imported capture file.
/alias	<i>string</i>	The alternative alias optionally manually defined via the Web Interface for this capture source.
/name	<i>string</i>	The name used for the capture instance in the Web Interface and logging if no alias is defined.
/encType	<i>string</i>	The type of encryption used for remote capture instances. Valid values: <i>none</i> , <i>aes-256</i> .
/encKey	<i>string</i>	The encryption key used for remote capture instances.
/host	<i>string</i>	The configured host for remote capture instances.
/port	<i>integer</i>	The configured port for remote capture instances.
/type	<i>string</i>	The type of capture instance. Valid values: <i>interface</i> (for local network interface), <i>remote</i> (for capture on a remote instance), <i>file</i> (for imported capture file).

12.2.1 /api/captures/management

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

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

12.3 /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)
-----> /dashcam
-----> /enabled (boolean)
-----> /duration (integer)
-----> /url (string)
-----> /interfaces
-----> /[sequential index number]
-----> /alias (string)
-----> /name (string)
-----> /segmentID (integer)
-----> /subdomainFilter (string)
-----> /domainFilter (integer)
-----> /protocolFilter (string)
-----> /versionFilter (string)
-----> /management
-----> /enabled (boolean)
-----> /subdomain (string)
-----> /domain (integer)
-----> /interval (integer)
-----> /ipv6MulticastScope (string)
-----> /loopback (boolean)
-----> /protocol (string)
-----> /version (string)
-----> /remoteConnections
-----> /[sequential index number]
-----> /alias (string)
-----> /host (string)
-----> /port (integer)
-----> /segmentID (string)
-----> /encType (string)
-----> /encKey (string)
-----> /evaluation
-----> /enabled (boolean)

```

* Directly referenceable endpoint


```

-> /api/config*
-----> /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)
-----> /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)
-----> /events (string array)
-----> /license
-----> /id (string)
-----> /key (string)
-----> /user (string)

```

* Directly referenceable endpoint

```

-> /api/config*
-----> /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)
-----> /terminology
-----> /states
-----> /[sequential index number]
-----> /value (string)
-----> /display (string)
-----> /users
-----> /[sequential index number]
-----> /username (string)
-----> /password (string)
-----> /salt (string)
-----> /writeAccess (boolean)

```

* Directly referenceable endpoint

12.3.1 /api/config/api

12.3.1.1 /api/config/api/fileServer

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

12.3.1.2 /api/config/api/http

/enabled	<i>boolean</i>	Specifies if HTTP access is enabled.
/port	<i>integer</i>	Specifies the port for HTTP access.

12.3.1.3 /api/config/api/https

/enabled	<i>boolean</i>	Specifies if HTTPS access is enabled.
/cert	<i>string</i>	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	<i>string</i>	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	<i>integer</i>	Specifies the port for HTTPS access.

12.3.2 /api/config/capture

/autoDump	<i>boolean</i>	Specifies whether auto-dump is enabled.
/initDuration	<i>integer</i>	Specifies the initialization wait period in seconds before events are generated.

12.3.2.1 /api/config/capture/dashcam

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

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

/alias	<i>string</i>	The alias manually specified for each network interface.
/name	<i>string</i>	Specifies the interface name as set by Npcap (Windows) or the operating system (Linux/Mac).
/segmentID	<i>integer</i>	Specifies the segment ID which traffic captured via this interface will be assigned to.
/subdomainFilter	<i>string</i>	Specifies the PTPv1 subdomain name if capture traffic via this interface is to be limited to that subdomain. If set to <i>any</i> , management messages are sent to all commonly used subdomains (<i>_DFLT</i> , <i>_ALT1</i> , <i>_ALT2</i> , <i>_ALT3</i>)
/domainFilter	<i>integer</i>	Specifies the PTPv2/PTPv2.1 domain number if capture traffic via this interface is to be limited to that domain.
/protocolFilter	<i>string</i>	Specifies the network protocol if capture traffic via this interface is to be limited to that protocol.
/versionFilter	<i>string</i>	Specifies the PTP version if capture traffic is to be limited to that version.

/management

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

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

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

12.3.3 /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.

12.3.4 /api/config/event

12.3.4.1 /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/*

/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.

12.3.4.2 /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.

<code>/version</code>	<i>string</i>	The SNMP version used by the SNMP manager. Valid values: <i>SNMPv1</i> , <i>SNMPv2c</i> , <i>SNMPv3</i> .
<code>/authentication/enabled</code>	<i>boolean</i>	Specifies whether the SMTP smarthost requires authentication. Only applies to SMTP servers.
<code>/authentication/username</code>	<i>string</i>	The SMTP smarthost authentication username. Only applies to SMTP servers.
<code>/authentication/password</code>	<i>string</i>	The SMTP smarthost authentication password. Only applies to SMTP servers.

12.3.5 `/api/config/license`

<code>/id</code>	<i>string</i>	The registered license ID.
<code>/key</code>	<i>string</i>	The registered license key.
<code>/user</code>	<i>string</i>	The registered license name.

12.3.6 `/api/config/logging`

12.3.6.1 `/api/config/logging/file`

<code>/enabled</code>	<i>boolean</i>	Specifies whether logging to file output is enabled.
<code>/filename</code>	<i>string</i>	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.
<code>/severity</code>	<i>string</i>	The maximum severity of log output to file. Valid values: <i>error</i> , <i>warning</i> , <i>info</i> , <i>debug</i> , <i>trace</i> .

12.3.6.2 `/api/config/logging/standardStreams`

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

12.3.6.3 `/api/config/logging/syslog`

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

12.3.7 `/api/config/memory`

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

12.3.8 /api/config/remote

/enabled	<i>boolean</i>	Specifies whether the outbound transmission of capture data is enabled.
/port	<i>integer</i>	The port over which outbound capture data is sent.

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

12.3.9 /api/config/terminology

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

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

12.3.10 /api/config/users/[_id]

/username	<i>string</i>	The username of the user.
/password	<i>string</i>	The hashed password of the user.
/salt	<i>string</i>	The string used to salt the hashed password of the user in transit.
/writeAccess	<i>boolean</i>	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.

12.4 /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)*
-----> /delayMechanism (string)*
-----> /device (integer)*
-----> /frequencyTraceable (boolean)*
-----> /grandmaster (integer)*
-----> /grandmasterDevice (integer)*
-----> /id (integer)*
-----> /illegalMaster (boolean)*
-----> /leap59 (boolean)*
-----> /leap61 (boolean)*
-----> /parent (integer)*
-----> /parentDevice (integer)*
-----> /port (integer)*
-----> /scope (integer)*
-----> /state (string)*
-----> /stepsRemoved (integer)*
-----> /timeSource (string)*
-----> /timeTraceable (boolean)*
-----> /timescale (boolean)*
-----> /timestampMechanism (string)*
-----> /utcOffset (integer)*
-----> /_links*
-----> /device (string)
-----> /grandmaster (string)
-----> /grandmasterDevice (string)
-----> /scope (string)
-----> /self (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)*

```

* Directly referenceable endpoint

```

-> /api/current*
-----> /instances*
-----> /[sequential index number]*
-----> /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)*
-----> /total (integer)*
-----> /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)*

```

* 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)*
-----> /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)*
-----> /inStore (integer)*
-----> /total (integer)*
-----> /segments*
-----> /[sequential index number]*
-----> /_index (integer)
-----> /id (integer)*
-----> /name (string)*
-----> /description (string)*
-----> /_links*
-----> /self (string)

```

* Directly referenceable endpoint

12.4.1 /api/current/devices/[_id]

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

12.4.1.1 /api/current/devices/[_id]/_links

/image	<i>string</i>	The image set for the device to be displayed in the devices map. This is relative to the file server path <i>/api/config/api/fileServer/directory</i> .
/self	<i>string</i>	The API endpoint to the device itself under <i>/api/current/devices</i> .
/vendorImage	<i>string</i>	The image set for the device vendor. This is relative to the file server path <i>/api/config/api/fileServer/directory</i> .

12.4.1.2 /api/current/devices/[_id]/custom

/alias	<i>string</i>	The manually set alias for the device. Overrides the automatically acquired alias.
/firmwareRevision	<i>string</i>	The manually set firmware revision for the device. Overrides the automatically acquired firmware revision.
/hardwareRevision	<i>string</i>	The manually set hardware revision for the device. Overrides the automatically acquired hardware revision.
/softwareRevision	<i>string</i>	The manually set software revision for the device. Overrides the automatically acquired software revision.
/imagePath	<i>string</i>	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	<i>string</i>	The manually set physical location for the device. Overrides the automatically acquired physical location.
/modelName	<i>string</i>	The manually set model name for the device. Overrides the automatically acquired model name.
/vendor	<i>string</i>	The manually set vendor for the device. Overrides the automatically acquired vendor name.

12.4.2 /api/current/instances/[_id]

/_index	<i>integer</i>	The sequentially assigned index value for the instance.
/address	<i>string</i>	The IPv4, IPv6, or MAC address of the instance.
/announceReceiptTimeout	<i>integer</i>	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.
/delayMechanism	<i>string</i>	The delay mechanism used by the instance for measuring propagation delays. Valid values: <i>E2E</i> , <i>P2P</i>
/device	<i>integer</i>	The device ID (<i>/api/current/devices/*id</i>) to which this instance is bound.
/frequencyTraceable	<i>boolean</i>	Specifies whether the frequency signal can be traced back to its primary reference.
/grandmaster	<i>integer</i>	Specifies the instance ID (<i>/api/current/instances/*id</i>) 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 <i>/id</i> below).
/grandmasterDevice	<i>integer</i>	Specifies the device ID (<i>/api/current/devices/*id</i>) 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 <i>/device</i> above).
/id	<i>integer</i>	The ID set by the PTP Track Hound v2 capture service for this instance.
/illegalMaster	<i>boolean</i>	Specifies if this instance is an illegal master.

<code>/leap59</code>	<i>boolean</i>	Specifies if the removal of a leap second is scheduled for this instance.
<code>/leap61</code>	<i>boolean</i>	Specifies if the insertion of a leap second is scheduled for this instance.
<code>/parent</code>	<i>integer</i>	Specifies the ID of the parent instance (<code>/api/current/instances/*/<i>id</i></code>) of this instance. If this instance is a top-level instance, this value will be <i>null</i> .
<code>/parentDevice</code>	<i>integer</i>	Specifies the ID of the parent instance's device (<code>/api/current/devices/*/<i>id</i></code>) for this instance. If this instance is a top-level instance, this value will be <i>null</i> .
<code>/port</code>	<i>integer</i>	The port number of the instance; this is conventionally appended to the clockID.
<code>/scope</code>	<i>integer</i>	The scope ID that this instance has been assigned to by PTP Track Hound v2.
<code>/state</code>	<i>string</i>	The port state of this instance. The strings here reference the terminology defined at <code>/api/config/terminology/states/*/<i>value</i></code>
<code>/stepsRemoved</code>	<i>integer</i>	The number of steps that this instance is removed from its grandmaster. Will be <i>0</i> if the instance is itself the grandmaster.
<code>/timeSource</code>	<i>string</i>	The primary time reference for this instance as provided via <i>Announce</i> or <i>Management</i> messages, if known. Valid values: <i>Atomic Clock, GPS, Terrestrial Radio, PTP, NTP, Hand-Set, Other, Internal Oscillator, Unknown</i>
<code>/timeTraceable</code>	<i>boolean</i>	Specifies whether the time can be traced back to its primary reference.
<code>/timescale</code>	<i>boolean</i>	If <i>true</i> , this instance is using the PTP timescale. If <i>false</i> , this instance is using an arbitrary timescale.
<code>/timestampMechanism</code>	<i>string</i>	Specifies the timestamping mechanism of this instance. Valid values: <i>One-Step, Two-Step</i>
<code>/utcOffset</code>	<i>integer</i>	Specifies the offset between PTP time and UTC time in seconds.

12.4.2.1 `/api/current/instances/[_id]/_links`

<code>/device</code>	<i>string</i>	The API endpoint to the device on which the instance is running.
<code>/grandmaster</code>	<i>string</i>	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.
<code>/grandmasterdevice</code>	<i>string</i>	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.
<code>/scope</code>	<i>string</i>	The API endpoint of the scope to which this instance has been assigned.
<code>/self</code>	<i>string</i>	This instance's own API endpoint as a link.

12.4.2.2 /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 <i>Announce</i> message interval of the instance.
/delayReq	string	Specifies the <i>Delay Request</i> message interval of the instance.
/delayResp	string	Specifies the <i>Delay Response</i> message interval of the instance.
/followUp	string	Specifies the <i>Follow-Up</i> message interval of the instance.
/management	string	Specifies the <i>Management</i> message interval of the instance.
/monitoringReq	string	Specifies the <i>Monitoring Request</i> message interval of the instance.
/monitoringResp	string	Specifies the <i>Monitoring Response</i> message interval of the instance.
/pDelayReq	string	Specifies the <i>Peer Delay Request</i> message interval of the instance.
/pDelayResp	string	Specifies the <i>Peer Delay Response</i> message interval of the instance.
/pDelayRespFollowUp	string	Specifies the <i>Peer Delay Response Follow-Up</i> message interval of the instance.
/signalling	string	Specifies the <i>Signaling</i> message interval of the instance.
/sync	string	Specifies the <i>Sync</i> message interval of the instance.

12.4.2.3 /api/current/instances/[_id]/packetCount

/announce	integer	Specifies the total number of <i>Announce</i> messages counted for this instance since last erasure.
/delayReq	integer	Specifies the total number of <i>Delay Request</i> messages counted for this instance since last erasure.
/delayResp	integer	Specifies the total number of <i>Delay Response</i> messages counted for this instance since last erasure.
/followUp	integer	Specifies the total number of <i>Follow-Up</i> messages counted for this instance since last erasure.
/management	integer	Specifies the total number of <i>Management</i> messages counted for this instance since last erasure.
/monitoringReq	integer	Specifies the total number of <i>Monitoring Request</i> messages counted for this instance since last erasure.
/monitoringResp	integer	Specifies the total number of <i>Monitoring Response</i> messages counted for this instance since last erasure.
/pDelayReq	integer	Specifies the total number of <i>Peer Delay Request</i> messages counted for this instance since last erasure.
/pDelayResp	integer	Specifies the total number of <i>Peer Delay Response</i> messages counted for this instance since last erasure.

<code>/pDelayRespFollowUp</code>	<i>integer</i>	Specifies the total number of <i>Peer Delay Response Follow-Up</i> messages counted for this instance since last erasure.
<code>/signalling</code>	<i>integer</i>	Specifies the total number of <i>Signaling</i> messages counted for this instance since last erasure.
<code>/sync</code>	<i>integer</i>	Specifies the total number of <i>Sync</i> messages counted for this instance since last erasure.
<code>/total</code>	<i>integer</i>	Specifies the total number of PTP messages counted for this instance since last erasure.

12.4.2.4 `/api/current/instances/[_id]/grandmasterQuality`

<code>/clockAccuracy</code>	<i>string</i>	The accuracy of the Grandmaster according to the PTP dataset, followed by the IEEE 1588 accuracy state code as a hexadecimal value in parentheses.
<code>/clockAccuracyText</code>	<i>string</i>	The accuracy of the Grandmaster according to the PTP dataset, without the IEEE 1588 accuracy state code.
<code>/clockAccuracyValue</code>	<i>integer</i>	The accuracy of the Grandmaster according to the PTP dataset, as the IEEE 1588 accuracy state in decimal.
<code>/clockClass</code>	<i>integer</i>	The Clock Class of the Grandmaster according to the PTP dataset.
<code>/clockID</code>	<i>string</i>	The Clock ID of the Grandmaster according to the PTP dataset.
<code>/clockVariance</code>	<i>string</i>	The Clock Variance of the Grandmaster according to the PTP dataset.
<code>/priority1</code>	<i>integer</i>	The Priority 1 value of the Grandmaster according to the PTP dataset.
<code>/priority2</code>	<i>integer</i>	The Priority 2 value of the Grandmaster according to the PTP dataset.
<code>/valid</code>	<i>boolean</i>	Specifies whether the dataset of the Grandmaster is valid.

12.4.2.5 `/api/current/instances/[_id]/localQuality`

<code>/clockAccuracy</code>	<i>string</i>	The accuracy of the clock according to the PTP dataset, followed by the IEEE 1588 accuracy state code as a hexadecimal value in parentheses.
<code>/clockAccuracyText</code>	<i>string</i>	The accuracy of the clock according to the PTP dataset, without the IEEE 1588 accuracy state code.
<code>/clockAccuracyValue</code>	<i>integer</i>	The accuracy of the clock according to the PTP dataset, as the IEEE 1588 accuracy state code in decimal.
<code>/clockClass</code>	<i>integer</i>	The Clock Class of the clock according to the PTP dataset.
<code>/clockID</code>	<i>string</i>	The Clock ID of the clock according to the PTP dataset.
<code>/clockVariance</code>	<i>string</i>	The Clock Variance of the clock according to the PTP dataset.
<code>/priority1</code>	<i>integer</i>	The Priority 1 value of the clock according to the PTP dataset.
<code>/priority2</code>	<i>integer</i>	The Priority 2 value of the clock according to the PTP dataset.
<code>/valid</code>	<i>boolean</i>	Specifies whether the dataset of the clock is valid.

12.4.3 /api/current/packets

/latest	<i>integer</i>	Specifies the ID number of the latest PTP packet to be captured.
/oldest	<i>integer</i>	Specifies the ID number of the oldest PTP packet currently in the capture log.

12.4.3.1 /api/current/packets/count

/announce	<i>integer</i>	Specifies the total number of <i>Announce</i> messages counted since last erasure.
/delayReq	<i>integer</i>	Specifies the total number of <i>Delay Request</i> messages counted since last erasure.
/delayResp	<i>integer</i>	Specifies the total number of <i>Delay Response</i> messages counted since last erasure.
/followUp	<i>integer</i>	Specifies the total number of <i>Follow-Up</i> messages counted since last erasure.
/management	<i>integer</i>	Specifies the total number of <i>Management</i> messages counted since last erasure.
/monitoringReq	<i>integer</i>	Specifies the total number of <i>Monitoring Request</i> messages counted since last erasure.
/monitoringResp	<i>integer</i>	Specifies the total number of <i>Monitoring Response</i> messages counted since last erasure.
/pDelayReq	<i>integer</i>	Specifies the total number of <i>Peer Delay Request</i> messages counted since last erasure.
/pDelayResp	<i>integer</i>	Specifies the total number of <i>Peer Delay Response</i> messages counted since last erasure.
/pDelayRespFollowUp	<i>integer</i>	Specifies the total number of <i>Peer Delay Response Follow-Up</i> messages counted since last erasure.
/signalling	<i>integer</i>	Specifies the total number of <i>Signaling</i> messages counted since last erasure.
/sync	<i>integer</i>	Specifies the total number of <i>Sync</i> messages counted since last erasure.
/v1	<i>integer</i>	Specifies the total number of incoming PTPv1 messages counted since last erasure.
/v2	<i>integer</i>	Specifies the total number of incoming PTPv2 and PTPv2.1 messages counted since last erasure.
/v2_0	<i>integer</i>	Specifies the total number of incoming PTPv2 (not PTPv2.1) messages counted since last erasure.
/v2_1	<i>integer</i>	Specifies the total number of incoming PTPv2.1 messages counted since last erasure.

<code>/total</code>	<i>integer</i>	Specifies the total number of PTP messages counted since last erasure.
<code>/inStore</code>	<i>integer</i>	Specifies the total number of PTP messages currently held in the capture log.
<code>/bytes</code>	<i>integer</i>	Specifies the total content of PTP packets counted since last erasure in bytes.

12.4.3.2 `/api/current/packets/perSecond`

For more information on how these per-second values are calculated, please refer to Chapter 8, "Traffic".

<code>/announce</code>	<i>float</i>	Specifies the current number of <i>Announce</i> messages received per second.
<code>/delayReq</code>	<i>float</i>	Specifies the current number of <i>Delay Request</i> messages received per second.
<code>/delayResp</code>	<i>float</i>	Specifies the current number of <i>Delay Response</i> messages received per second.
<code>/followUp</code>	<i>float</i>	Specifies the current number of <i>Follow-Up</i> messages received per second.
<code>/management</code>	<i>float</i>	Specifies the current number of <i>Management</i> messages received per second.
<code>/monitoringReq</code>	<i>float</i>	Specifies the current number of <i>Monitoring Request</i> messages received per second.
<code>/monitoringResp</code>	<i>float</i>	Specifies the current number of <i>Monitoring Response</i> messages received per second.
<code>/pDelayReq</code>	<i>float</i>	Specifies the current number of <i>Peer Delay Request</i> messages received per second.
<code>/pDelayResp</code>	<i>float</i>	Specifies the current number of <i>Peer Delay Response</i> messages received per second.
<code>/pDelayRespFollowUp</code>	<i>float</i>	Specifies the current number of <i>Peer Delay Response Follow-Up</i> messages received per second.
<code>/signalling</code>	<i>float</i>	Specifies the current number of <i>Signaling</i> messages received per second.
<code>/sync</code>	<i>float</i>	Specifies the current number of <i>Sync</i> messages received per second.
<code>/time</code>	<i>float</i>	The time at which the last per-second values were calculated as a Unix timestamp.
<code>/total</code>	<i>float</i>	Specifies the current number of PTP messages in general received per second.
<code>/v1</code>	<i>float</i>	Specifies the current number of PTPv1 messages received per second.

<code>/v2</code>	<i>float</i>	Specifies the current number of PTPv2 and PTPv2.1 messages received per second.
<code>/v2_0</code>	<i>float</i>	Specifies the current number of PTPv2 (not PTPv2.1) messages received per second.
<code>/v2_1</code>	<i>float</i>	Specifies the current number of PTPv2.1 messages received per second.
<code>/bytes</code>	<i>float</i>	Specifies the current data rate of incoming PTP packets per second in bytes.

12.4.4 `/api/current/scopes/[_id]`

<code>/_index</code>	<i>integer</i>	The sequentially assigned index value for the scope.
<code>/id</code>	<i>integer</i>	The unique ID of the scope assigned by PTP Track Hound v2.
<code>/domain</code>	<i>integer</i> or <i>string</i>	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.
<code>/protocol</code>	<i>string</i>	The common network protocol employed by all instances in this scope. Valid values: <i>IPv4</i> , <i>IPv6</i> , <i>IEEE 802.3</i>
<code>/version</code>	<i>string</i>	The common PTP version employed by all instances in this scope. Valid values: <i>PTPv1</i> , <i>PTPv2</i> , <i>PTPv2.1</i>
<code>/compatibility</code>	<i>boolean</i>	If <i>true</i> , 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 <i>false</i> , this is a PTPv2.1 that only contains PTPv2.1 instances. Accordingly, this resource does not appear in non-PTPv2.1 scopes.
<code>/vlanID</code>	<i>string</i>	The common VLAN tag employed by all instances in this scope. If VLAN tagging is not used, this will be <i>none</i> .
<code>/segmentID</code>	<i>integer</i>	The manually defined segment ID to which all instances in this scope belong.

12.4.4.1 `/api/current/scopes/[_id]/_links`

<code>/self</code>	<i>string</i>	This scope's own API endpoint as a link.
--------------------	---------------	--

12.4.4.2 /api/current/scopes/[_id]/packetCount

/announce	<i>integer</i>	Specifies the total number of <i>Announce</i> messages counted for this scope since last erasure.
/delayReq	<i>integer</i>	Specifies the total number of <i>Delay Request</i> messages counted for this scope since last erasure.
/delayResp	<i>integer</i>	Specifies the total number of <i>Delay Response</i> messages counted for this scope since last erasure.
/followUp	<i>integer</i>	Specifies the total number of <i>Follow-Up</i> messages counted for this scope since last erasure.
/management	<i>integer</i>	Specifies the total number of <i>Management</i> messages counted for this scope since last erasure.
/monitoringReq	<i>integer</i>	Specifies the total number of <i>Monitoring Request</i> messages counted for this scope since last erasure.
/monitoringResp	<i>integer</i>	Specifies the total number of <i>Monitoring Response</i> messages counted for this scope since last erasure.
/pDelayReq	<i>integer</i>	Specifies the total number of <i>Peer Delay Request</i> messages counted for this scope since last erasure.
/pDelayResp	<i>integer</i>	Specifies the total number of <i>Peer Delay Response</i> messages counted for this scope since last erasure.
/pDelayRespFollowUp	<i>integer</i>	Specifies the total number of <i>Peer Delay Response Follow-Up</i> messages counted for this scope since last erasure.
/signalling	<i>integer</i>	Specifies the total number of <i>Signaling</i> messages counted for this scope since last erasure.
/sync	<i>integer</i>	Specifies the total number of <i>Sync</i> messages counted for this scope since last erasure.
/inStore	<i>integer</i>	Specifies the total number of PTP messages currently held in the capture log for this scope.
/total	<i>integer</i>	Specifies the total number of PTP messages counted for this scope since last erasure.

12.4.5 /api/current/segments/[_id]

/_index	<i>integer</i>	The sequentially assigned index value for the segment.
/id	<i>integer</i>	The unique ID of the segment assigned by PTP Track Hound v2.
/name	<i>string</i>	The manually entered name of the segment that appears in the Web Interface as the heading.
/description	<i>string</i>	The manually entered description for the segment that appears in the Web Interface beneath the heading.

12.4.5.1 /api/current/segments/{_id}/_links

/self *string* This segment's own API endpoint as a link.

12.5 /api/events/[_id]

The *events* resources provide information about logged events in PTP Track Hound v2.

```
-> /api/events*
-----> /[sequential index number]
-----> /_index (integer)
-----> /event (string)
-----> /brief (string)
-----> /details (string)
-----> /id (integer)
-----> /severity (string)
-----> /timestamp (string)
-----> /time
-----> /seconds (integer)
-----> /microseconds (integer)
```

* Directly referenceable endpoint

/_index	<i>integer</i>	The sequential index number assigned in the resource level above.
/event	<i>string</i>	The type of event. Valid values: <i>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</i>
/brief	<i>string</i>	A brief description of the event.
/details	<i>string</i>	Where appropriate, additional details regarding the event (such as dataset changes) are provided here.
/id	<i>integer</i>	The ID set by the PTP Track Hound v2 capture service for this event.
/severity	<i>string</i>	The severity of this event; this relates to the log levels of the various logging options.
/timestamp	<i>string</i>	The timestamp of the event in the format <i>YYYY-MM-DD'THH:MM:SS.SSSSSS</i> .

12.5.1 /api/events/[_id]/time

/seconds	<i>integer</i>	The event timestamp in UNIX time (whole seconds). Corresponds to the Gregorian timestamp under <i>/api/events/*/timestamp</i> .
/microseconds	<i>integer</i>	The number of microseconds after the last whole second in UNIX time of the event. Corresponds to the Gregorian timestamp under <i>/api/events/*/timestamp</i> .

12.6 /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	<i>string</i>	The build date and time of this version of PTP Track Hound v2 in the format YYYY-MM-DD'T'HH:MM:SS
/name	<i>string</i>	The name of the software. Always returns "PTP Track Hound".
/platform	<i>string</i>	The name of the operating system on which PTP Track Hound v2 is running. Valid values: <i>Windows, Linux, macOS</i>
/running	<i>boolean</i>	Is <i>true</i> if the PTP Track Hound v2 capture service is running, or if a capture file is currently open and being analyzed.
/version	<i>string</i>	The PTP Track Hound v2 version.

12.6.1 /api/info/license

/id	<i>string</i>	The ID code of the currently registered license.
/key	<i>string</i>	The authentication key of the currently registered license.
/type	<i>string</i>	The license level of the currently registered license. While the valid values in theory are <i>Free, Basic, and Professional</i> , the REST API requires a Professional License to access, so that in practice, this resource will only ever be readable as <i>Professional</i> via the REST API.
/user	<i>string</i>	The name of the user associated with the currently registered license.
/version	<i>string</i>	The PTP Track Hound version to which this license applies.
/expiration	<i>string</i>	This resource provides the expiry date of the license in the format YYYY-MM-DD if the license is time-limited.

12.7 /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	<i>integer</i>	The currently configured size of each allocable memory chunks in bytes. Identical to <i>/api/config/memory/chunkSize</i> .
/maxBytes	<i>integer</i>	The currently configured maximum amount of memory in bytes usable by PTP Track Hound v2 for captured data, analysis, and modeling. Identical to <i>/api/config/memory/max</i> .

12.7.1 /api/memory/used

/data	<i>integer</i>	The amount of memory in bytes currently used by the PTP Track Hound v2 service for the storage of capture data.
/model	<i>integer</i>	The amount of memory in bytes currently used by the PTP Track Hound v2 service for data modeling.
/total	<i>integer</i>	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. <i>/api/memory/maxBytes</i> represents the maximum that this value can reach.

12.8 /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*
-----> /[sequential index number]
-----> /description (string)
-----> /friendlyName (string)
-----> /index (integer)
-----> /ipAddresses (string array)
-----> /linkState (string)
-----> /macAddress (string)
-----> /name (string)
```

* Directly referenceable endpoint

12.8.1 /api/network/interfaces/[_id]

<code>/index</code>	<i>integer</i>	The index number assigned by the operating system to this physical network interface.
<code>/ipAddresses</code>	<i>string array</i>	An array containing the IPv4 and IPv6 addresses of the virtual interfaces assigned to this physical network interface.
<code>/linkState</code>	<i>string</i>	The physical link state of this physical network interface. Valid values: <i>up</i> , <i>down</i> .
<code>/macAddress</code>	<i>string</i>	The MAC address of this physical network interface.
<code>/name</code>	<i>string</i>	The name of the network interface as assigned by the packet capture service.
<code>/friendlyName</code>	<i>string</i>	The friendly name of the network interface as assigned by the operating system. Only applicable to Windows.
<code>/description</code>	<i>string</i>	The description of the network interface as assigned by the operating system. Only applicable to Windows.

12.9 /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

12.9.1 /api/files/capture

<code>/apiPath</code>	<i>string</i>	The API endpoint referencing this capture file.
<code>/filename</code>	<i>string</i>	The filename (without the path) of this capture file.
<code>/localPath</code>	<i>string</i>	The full path to the capture file, relative to the local device on which PTP Track Hound v2 is running.
<code>/size</code>	<i>integer</i>	The size of the file in bytes.

12.9.2 /api/files/img

<code>/apiPath</code>	<i>string</i>	The API endpoint referencing this image file.
<code>/filename</code>	<i>string</i>	The filename (without the path) of this image file.
<code>/localPath</code>	<i>string</i>	The full path to the image file, relative to the local device on which PTP Track Hound v2 is running.
<code>/size</code>	<i>integer</i>	The size of the file in bytes.

13 Appendix

13.1 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 the 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.