Tanium™ Client Recorder Extension User Guide

Version 1.0.0

August 12, 2019

Table of contents

Client Recorder Extension overview

The Client Recorder Extension is a feature common to the Tanium Integrity Monitor, Tanium Map, Tanium Threat Response, and Tanium Trace solution modules. It continuously saves key forensic evidence on each endpoint. The Client Recorder Extension monitors the endpoint kernel and other low-level subsystems to capture a variety of events.

Traditional disk and memory forensics techniques can successfully reconstruct fragments of endpoint activity, but are limited to the evidence that is natively preserved by the underlying operating system. This type of evidence from a period of interest can rapidly degrade as time elapses. In contrast, the Client Recorder Extension maintains a complete, easy-to-interpret history of events so you can replay recent system events.

Even an idle system quickly accumulates data. The Client Recorder Extension stores event data in a local database. The default configuration can retain up to several months of historical data. You can customize the amount of local storage that is consumed by the Client Recorder Extension, and filter the types of recorded evidence.

Types of recorded events

The Client Recorder Extension captures a broad range of events, that include additional context and metadata. Recorded event examples include:

- process execution
- file system activity
- registry changes
- network connections
- driver and library loads
- user authentication

You can specify which process, registry, network, file, and security events to record, depending on whether or not they apply to the operating systems of the endpoints.

> **Tip:** For more meaningful databases and to retain data for longer periods, consider excluding events that occur frequently; for example, LanguageList registry values are a verbose event on Windows endpoints.

You can configure filters to limit event recording to selected registry events, network events, file events, security events, or DNS events.

REGISTRY

[Windows only] Changes to the registry, such as the creation or alteration of registry keys and values. Includes the associated process and user context.

NETWORK

Network connection events, such as an HTTP request to an internet location, including the associated process and user context. Events are recorded for all inbound and outbound TCP connections.

FILE

File system events, such as files written to directory locations on the endpoint. The associated process and user context are included. Examples: A malware file copied to a location that Windows Update uses, or content changes made to a file.

SECURITY

[Windows and Linux only] Security events such as authentication, privilege escalation, and more. This event type includes logon events.

DNS

[Windows 8.1 or later] Request information, including the process path, user, query, response, and the type of operation.

Sources of Client Recorder Extension data on Windows

The Client Recorder Extension gathers data from multiple sources into a single, local database on the endpoint. Kernel events are gathered from Windows tools. On Windows endpoints, the optional Microsoft Sysmon configuration provides additional information about the executed processes.

Some features of the Client Recorder Extension require specific versions of Windows.

Table 1: Client Recorder Extension features - Windows

Feature	Windows Server 2008 R2	Windows Server 2012	Windows Server 2012 R2 or later	Windows 7	Windows 8	Windows 8.1 or later
DNS events	Not Available	Not Available	Available	Not Available	Not Available	Available

Feature	Windows Server 2008 R2	Windows Server 2012	Windows Server 2012 R2 or later	Windows 7	Windows 8	Windows 8.1 or later
Process hashes and command-line information	Requires Tanium driver or Sysmon	Tanium driver recommended	Tanium driver recommended	Requires Tanium driver or Sysmon	Tanium driver recommended	Tanium driver recommended
Driver loads	Available*	Available	Available	Available*	Available	Available

* If Sysmon is configured, the driver load information recorded by Sysmon is used.

Sources of Client Recorder Extension data on Linux

The Client Recorder Extension for Linux uses the Linux audit subsystem to collect events. The Client Recorder Extension for Linux uses the following components for event collection:

Kernel Driver (kaudit)

This process is a part of the Linux kernel responsible for the kernel audit events and will forward audited events to the uauditd process. Audited events are defined by a rules file. This rules file is called audit.rules and is located at `/etc/audit/audit.rules`. Additional rules files that can be read into the kauditd process and added to the `audit.rules` file are located in `/etc/audit/rules.d/`.

Audit Daemon (auditd)

This process communicates to the to the kernel via the netlink socket. For most Linux versions this is limited to a single listener. This process writes to audit log files or forwards events to the audispd process for dispatching.

Audit Dispatcher (audispd)

This process is an event multiplexor that helps overcome limitations of single listener socket. This process consumes audit events from the auditd process and dispatches them to child plugins that want to analyze events in real-time. The

Client Recorder Extension is an example of one of these child plugins. The configuration for these child plugins is found under `/etc/audisp/plugins.d/`.

Client Recorder Extension

The Client Recorder Extension acts as an audispd plugin that collects audited events from the audispd process and writes them to a SQLite database named `monitor.db`. The tables in this database store raw events and are virtual tables that define a query of data from the recorded data tables to simplify data gathering. The Client Recorder Extension and `monitor.db` are located in `/opt/Tanium/TaniumClient/Tools/Trace/`.

Note: The recording of security and DNS events is not available on Linux.

Sources of Client Recorder Extension data on Mac

On Mac endpoints, the recorder collects data from the OpenBSM auditing system that is installed in all Mac releases from 10.8 to current.

The recorder connects to a clone of `/dev/auditpipe` to record events to monitor.db. When the recorder is installed on a Mac endpoint, `/etc/security/audit_class` is updated with a Tanium Recorder entry to map subscribed events at runtime. The recorder then clones `/dev/auditpipe` and configures the copy to use the `tan` audit class. When the recorder configuration is read, the types of wanted events are translated to the appropriate system calls to monitor, and those calls are then mapped to the tan audit class. This prevents the recorder from writing the audited events to the audit.log of the endpoint and allows the recorder to be very selective about which system calls are monitored.

The recorder writes the following event types to monitor.db:

Process Events

Command Lines of Process

Process Hashes

Network Events

File Events

Note: The recording of security and DNS events is not available on Mac.

This documentation may provide access to or information about content, products (including hardware and software), and services provided by third parties ("Third Party Items"). With respect to such Third Party Items, Tanium Inc. and its affiliates (i) are not responsible for such items, and expressly disclaim all warranties and liability of any kind related to such Third Party Items and (ii) will not be responsible for any loss, costs, or damages incurred due to your access to or use of such Third Party Items unless expressly set forth otherwise in an applicable agreement between you and Tanium.

Further, this documentation does not require or contemplate the use of or combination with Tanium products with any particular Third Party Items and neither Tanium nor its affiliates shall have any responsibility for any infringement of intellectual property rights caused by any such combination. You, and not Tanium, are responsible for determining that any combination of Third Party Items with Tanium products is appropriate and will not cause infringement of any third party intellectual property rights.

Getting started

1. Install the Client Recorder Extension. For more information, see Installing the Client Recorder Extension on page 14.
2. Configure endpoint settings. For more information, see Configuring endpoint settings on page 20.
3. Understand how multiple modules that use the Client Recorder Extension manage configuration settings. For more information, see Managing configurations across modules on page 50.

Client Recorder Extension requirements

Review the requirements before you install a module that includes the Client Recorder Extension.

Tanium dependencies

In addition to a license for a product module that contains the Client Recorder Extension, make sure that your environment also meets the following requirements.

Component	Requirement
Tanium Platform	6.5 or later. Enhanced functionality is available with version 7.0.314.6042 and later. Installing Tanium™ Interact is also suggested. For more information, see Tanium Core Platform Installation Guide: Installing Tanium Server.
Tanium Client	The Client Recorder Extension is supported on the same Linux and Mac endpoints as the Tanium Client. For Windows endpoints, you must have a minimum of Windows 7 or Windows Server 2008 R2. Windows 8.1 provides DNS event recording capability. For more information about specific Tanium Client versions, see Tanium Client Deployment Guide: Client host system requirements.
One of the following Tanium modules:	
Tanium Module	One of the following Tanium modules: • Tanium™ Trace • Tanium™ Threat Response • Tanium™ Integrity Monitor • Tanium™ Map

Tanium Module Server

Modules that install the Client Recorder Extension are installed and run as a service on the Module Server host computer. The impact on Module Server is minimal and depends on usage.

Endpoints

The Client Recorder Extension supports Windows, Linux, and Mac endpoints. For Windows endpoints, you must have a minimum of Windows 7 or Windows Server 2008 R2. Windows 8.1 provides DNS event recording capability. The amount of free disk space that is required depends on the configuration of the Client Recorder Extension. 3GB is recommended.

A minimum of 100 MB RAM is required on each endpoint device. By default, the endpoint database is 1 GB in size. There must be three times the maximum database size available in free disk space. The CPU demand on the endpoint averages less than 1%.

For Linux endpoints, you must:

- Install the most recent stable version of the audit daemon and audispd-plugins before initializing endpoints. See the specific operating system documentation for instructions.
- Be aware that when using immutable "-e 2" mode, the Client Recorder Extension adds Tanium audit rules in front of the immutable flag. When using the **-e 2** flag on Linux, the status sensor for each product that uses the Client Recorder Extension indicates if the recorder needs to be restarted.

You can use the Tanium Event Recorder Driver or Microsoft Sysmon to record process and command line events on supported Windows endpoints. The following operating systems support the Tanium Event Recorder Driver:

- Windows 7
- Windows Server 2008 R2
- Windows Server 2012
- Windows Server 2012 R2
- Windows 8.1
- Windows 10, build 1607 or later
- Windows Server 2016
- Windows Server 2019

Notes:

- Windows 7 and Windows Server 2008 R2 operating systems must have KB3033929 installed to ensure the Tanium signing certificates are trusted by the operating system. For details regarding KB3033929 , see https://support.microsoft.com/en-us/help/3033929/microsoft-security-

advisory-availability-of-sha-2-code-signing-support.

- Windows 10, build 1511, is not supported by the Tanium Event Recorder Driver.

Third-party software

(Windows, Optional) Microsoft Sysmon

In addition to the Tanium Event Recorder Driver, you can use the latest supported version of Microsoft Sysmon to record process hashes and command-line information on Windows endpoints earlier than Windows 8.1 and Windows Server 2012 R2. For Windows 8.1 or later and Windows Server 2012 R2 or later, Sysmon is not required.

Host and network security requirements

Security exclusions

If security software is in use in the environment to monitor and block unknown host system processes, your security administrator must create exclusions to allow the Tanium processes to run without interference. See the module user guide for a complete reference of exclusions that must be put in place for the module to work as expected. The following table lists the exclusions required for the Client Recorder Extension.

Target Device	Process
Module Server	`<Tanium Module Server>\services\<ProductName>\node.exe`
Endpoint computers (Windows)	`<Installation Location>\sysmon.exe`
Endpoint computers (Linux)	`<Tanium Client>/Tools/Trace/recorder`
Endpoint computers (Mac)	`<Tanium Client>/Tools/Trace/TaniumRecorder`

Installing the Client Recorder Extension

The Client Recorder Extension is installed by a module to record event data. The **Distribute Tools** packages that the Tanium platform uses distribute configuration files and software on all targeted endpoints. The following list details configuration files and software that the **Distribute Tools** package installs on endpoints for the modules that use the Client Recorder Extension.

Software and configuration files added by the Client Recorder Extension

`/opt/Tanium/TaniumClient/Tools/Trace/recorder` (Linux)
`/Library/Tanium/TaniumClient/Tools/Trace/TaniumRecorder` (Mac)
`C:\Program Files(x86)\Tanium\Tanium Client\extensions\TaniumCXTrace.dll` (Windows)

The Client Recorder Extension process. This process consumes events from the audispd process and writes them to `monitor.db`. On Windows endpoints, the Client Recorder Extension runs as a DLL that the Tanium Client uses.

Tanium Trace and Tanium Threat Response

`/opt/Tanium/TaniumClient/Tools/Trace/recorder.json` (Linux)
`/Library/Tanium/TaniumClient/Tools/Trace/recorder.json` (Mac)

The configuration file for the Client Recorder Extension. This file contains configuration values that are set in the module workbench. Configuration items include CPU Killswitch values, `monitor.db` size, the maximum number of days to store data, enabling or disabling auditd RAW logging, specifying a logging level, and configuring the path to `monitor.db` and `filters.json`.

`/opt/Tanium/TaniumClient/Tools/Trace/filters.json` (Linux)
`/Library/Tanium/TaniumClient/Tools/Trace/filters.json` (Mac)
`C:\Program Files (x86)\Tanium\Tanium Client\filters.json` (Windows)

A configuration file that contains filters for network, process, registry, and file events to filter from recording. These filters are configured from the Tanium Trace or Tanium Threat Response workbench.

`/etc/audisp/plugins.d/trace.conf` (Linux/Mac)

A configuration file for the audispd process to forward events to the Client Recorder Extension. This configuration file is also used to restart the Tanium Recorder when auditd is stopped or restarted.

`/opt/Tanium/TaniumClient/Tools/Trace/monitor.db` (Linux)
`/Library/Tanium/TaniumClient/Tools/Trace/monitor.db` (Mac)
`C:\Program Files (x86)\Tanium\Tanium Client\monitor.db` (Windows)

The database that the Client Recorder Extension creates. It contains a history of recorded event details.

Tanium Integrity Monitor

During the installation of Tanium Integrity Monitor, additional files are added to the Trace directory.

`/opt/Tanium/TaniumClient/Tools/Trace/im_recorder.json` (Linux)
`/Library/Tanium/TaniumClient/Tools/Trace/im_recorder.json` (Mac)
`HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\Tanium\Tanium Client\Trace` (Windows)

Integrity Monitor specific configuration for the Client Recorder Extension.

`/opt/Tanium/TaniumClient/Tools/Trace/im_filters.json` (Linux)
`/Library/Tanium/TaniumClient/Tools/Trace/im_filters.json` (Mac)
`C:\Program Files (x86)\Tanium\Tanium Client\im_filters.json` (Windows)

An Integrity Monitor specific configuration to define the types of events to record.

`/opt/Tanium/TaniumClient/Tools/Trace/watchlist.json` (Linux)
`/Library/Tanium/TaniumClient/Tools/Trace/watchlist.json` (Mac)
`C:\Program Files (x86)\Tanium\Tanium Client\watchlist.json` (Windows)

An Integrity Monitor specific configuration file that contains the list of paths, path exclusions, and file event operations to record. The values in this list override any filters found in `filters.json` from other Tanium modules.

Tanium Map

During the installation of Tanium Map additional files are added to the Trace directory.

`/opt/Tanium/TaniumClient/Tools/Trace/map_filters.json` (Linux)
`/Library/Tanium/TaniumClient/Tools/Trace/map_filters.json` (Mac)
`C:\Program Files (x86)\Tanium\Tanium Client\map_filters.json` (Windows)

The recorder filters file for Tanium Map.

`/opt/Tanium/TaniumClient/Tools/Trace/map_recorder.json` (Linux)
`/Library/Tanium/TaniumClient/Tools/Trace/map_recorder.json` (Mac)
`C:\Program Files (x86)\Tanium\Tanium Client\map_recorder.json` (Windows)

This configuration file stores configuration parameters for the Client Recorder Extension.

Configuration changes on endpoints

The **Distribute Tools** packages make changes to the audit configurations on the targeted endpoints when you install a module that uses the Client Recorder Extension.

The following list details changes to configuration files and the audit subsystem on Mac and Linux endpoints.

`/etc/audit/auditd.conf` (Linux)

A configuration file specific to the audit daemon. The Client Recorder Extension installation in the module workbench prompts an administrator to set RAW logging to enabled or disabled. The Client Recorder Extension creates a backup copy of the existing `auditd.conf` on the endpoint and names it `/etc/audit/auditd.conf.pretrace`.

`/etc/audisp/audispd.conf` (Linux)

A configuration file controls the configuration of the audit event dispatcher process. The Client Recorder Extension creates a backup copy of the existing `audispd.conf` file on the endpoint, and names it `/etc/audisp/audispd.conf.pretrace`. The Client Recorder Extension modifies the q_depth setting to 32768.

`/etc/audit/audit.rules` (Linux/Mac)

This file specifies the audit events that the kernel audit system logs. This file is loaded into the kernel audit system. When the Client Recorder Extension is installed on the endpoint a backup copy of the existing `audit.rules` file is created and named `/etc/audit/audit.rules.pretrace`.

Starting and stopping the Client Recorder Extension

You might need to manually start or stop the Client Recorder Extension. For example, if the size of `monitor.db` exceeded a set limit or if the CPU usage exceeded the threshold, which automatically disabled the Client Recorder Extension, you must resolve the underlying issue first and then manually restart the Client Recorder Extension. Or, if you find that the Client Recorder Extension is using more system resources than expected, you can stop the Client Recorder Extension and troubleshoot the issue with the risk of additional resource consumption.

IMPORTANT: After stopping, you must manually restart the Client Recorder Extension, as it does not restart automatically.

To stop the Client Recorder Extension, perform the steps that correspond to the appropriate operating system in the following table.

Operating System	Instructions
Windows	a. From the Windows Start Menu, click **Run**. b. Type `services.msc` and click **OK**. c. Locate the Tanium Client service, right-click and select **Stop**.
Linux	From the `Tools/<module>` directory, run the following script as root or superuser: `./recorder --stop`
Mac	From the `Tools/<module>` directory, run the following script as root or superuser: `./TaniumRecorder --stop`

To start the Client Recorder Extension perform the steps that correspond to the appropriate operating system in the following table.

Operating System	Instructions
Windows	a. From the Windows Start Menu, click **Run**. b. Type `services.msc` and click **OK**. c. Locate the Tanium Client service, right-click and select **Start**.
Linux	From the `Tools/<module>` directory, run the following script as root or superuser: `./recorder --start`
Mac	From the `Tools/Trace` directory, run the following script as root or superuser: `./TaniumRecorder --start`

You can also stop or start the recorder by deploying a package as an action.

1. Use a question to target the affected endpoints. For example, ask `Get Tanium Threat Response Status from all machines`.
2. Drill down to the specific endpoints.
3. Deploy the **Disable Tanium Recorder [*OS*]** package as an action to disable the recorder.
4. Deploy the **Enable Tanium Recorder [*OS*]**package as an action to enable the recorder.

(Optional) Install the Tanium Event Recorder Driver

You can install the Tanium Event Recorder Driver to more accurately capture process and command line events.

The Tanium Event Recorder Driver can exist on the same endpoints as Sysmon. If an endpoint has Sysmon, the Windows Event Recorder switches over from Sysmon to the Tanium Event Recorder Driver when the Tanium Client resets or when the endpoint reboots. You can force an endpoint to continue using Sysmon with the ForceSysmon=1 registry setting (DWORD in HKLM\Software\Wow6432\Tanium\Tanium Client\Trace\).

1. From the Main menu, ask the question `Get Tanium Driver Status from all machines` and click **Search**.
2. Select **Install Recommended**.
3. From the Deploy Action page, select **Install Tanium Driver**.
4. Validate successful installations by checking the validation query that runs at the end of the package installation.

5. Collect the action logs from any endpoints that fail the validation query using Live Response.
6. Run the action **Remove Tanium Driver** on any endpoints that return anything other than SERVICE_RUNNING for the Tanium Event Recorder Driver service status.

What to do next

See Getting started on page 10 for more information about using the Client Recorder Extension.

Configuring endpoint settings

Configure the Client Recorder Extension by defining settings in `recorder.json` (Linux/Mac), or as registry entries (Windows). Modules that use the Client Recorder Extension provide a default configuration. Consult with your Technical Account Manager (TAM) before changing any Client Recorder Extension configuration settings.

> **CAUTION:** Your TAM can advise you how to best configure the Client Recorder Extension for your purposes. Changing configuration parameters can have serious, and sometimes irrevocable consequences.

audisp

Defines the configuration and plugin of the audit event dispatcher. (Linux only)

audisp configuration

q_depth

A numeric value that specifies the size of the internal queue of the audit event dispatcher. A larger queue is capable of processing flood of events, but could hold events that are not processed when the audisp daemon is terminated. Increase this value in the event when syslog messages about dropped events are reported. Default: 80.

overflow_action

Specifies how the daemon should react to overflowing in the internal queue. When an overflow occurs, more events are being received than the queue can process. This parameter has the following choices:

- `ignore`
- `syslog`
- `suspend`
- `single`
- `halt`

If set to `ignore`, the audisp daemon does nothing. If set to `syslog`, a warning is issued to syslog. If set to `suspend`, the audisp daemon stops processing events but the daemon continues to be active. If set to `single`, the audisp daemon puts the endpoint in single user mode. If set to `halt`, the audisp daemon shuts down

the endpoint. Default: `SYSLOG`.

priority_boost

A non-negative number that specifies how much of a priority boost the audit event dispatcher should apply. This boost is in addition to the boost provided from the audit daemon. Default: 4. No change: 0.

max_restarts

A non-negative number that specifies the number of times the audit event dispatcher can attempt to restart a non responsive plugin. Default: 10.

name_format

Specifies how endpoint node names are inserted into the audit event stream. This parameter has the following choices:

- `none`
- `hostname`
- `fqd`
- `numeric`
- `user`

If set to `none`, no computer name is inserted into the audit event. If set to `hostname`, the name is returned by the gethostname syscall. If set to `fqd`, the hostname is resolved with DNS for a fully-qualified domain name of the endpoint. If set to `numeric`, the IP address of the endpoint is resolved. If set to `user`, an admin defined string from the name option returns. For more information on the name option, see the documentation for the name parameter. Default: `None`.

name

The admin defined string that identifies the endpoint if user is given as the name_ format option.

audisp plugin

active

Specifies to restart the Client Recorder Extension if auditd is started or restarted. Default: `Yes`.

direction

When enabled, this parameter provides insight to the event dispatcher about which direction events flow. Default: `In`.

path

Specifies the absolute path to the plugin executable. For internal plugins it corresponds to the name of the plugin.

type

Specifies the way in which the plugin executes to the dispatcher. Options are `builtin` and `always`. Use builtin for plugins that are internal to the audit event dispatcher, for example, `af_unix` and `syslog`. Use the `always` option for most if not all plugins. Defaut: `always`.

args

Specifies how arguments are passed to the child program. In most cases, plugins do not take arguments and use a configuration file to instruct how they should be configured. There is a limit of two arguments.

format

Options for this parameter are binary and string. Binary passes the data exactly as the audit event dispatcher gets it from the audit daemon. The string option instructs the dispatcher to completely change the event into a string suitable for parsing with the audit parsing library. Default: `string`.

auditd

The auditd section of the configuration file contains information specific to the audit daemon. It should contain one configuration keyword per line, an equal sign, and be followed by appropriate configuration information. (Linux only)

log_file

Specifies the full path name to the log file where audit records are stored. It must be a regular file and not a symlink.

log_format

Specifies how log information should be stored on disk. There are two options:

- `RAW`
- `NOLOG`

If set to `RAW`, audit records are stored in a format exactly as sent by the kernel. If this option is set to `NOLOG` all audit information is discarded, but does not affect data sent to the audit event dispatcher. Default: `RAW`

log_group

Specifies the group that is applied to permissions for the log file. The group name can be either numeric or spelled out. Default: `root`.

priority_boost

A non-negative number that instructs the audit daemon how much of a priority boost to assign. Default: 4. No change: 0.

flush

Specifies when to flush audit records. Values are:

- `none`
- `incremental`
- `data`
- `sync`

If set to `none`, no special effort is made to flush the audit records to disk. If set to `incremental`, the `freq` parameter specifies how often an explicit flush to disk is issued. If set to `data`, the audit daemon keeps the data portion of the disk file synchronized at all times. If set to `sync`, the audit daemon keeps both the data and meta-data fully synchronized with every write to disk.

freq

A non-negative number that instructs the audit damon the number of records to write before issuing an explicit flush to disk command. This value is only valid when the flush keyword is set to incremental.

num_logs

Specifies the number of log files to keep if `rotate` is provided as the `max_log_file_action`. If the number is less than two, logs are not rotated. This number must be 99 or less. As you increase the number of log files being rotated, you can

increment the kernel backlog setting as more time is required to rotate the files. Kernel backlog settings are typically specified in `/etc/audit/audit.rules`. Default: 0 (no rotation).

disp_qos

Specifies blocking/lossless or non-blocking/lossy communication between the audit daemon and the dispatcher. There is a 128k buffer between the audit daemon and dispatcher. If `lossy` is selected, incoming events to the dispatcher are discarded when this queue is full, but events are still written to disk if the log_ format is set to `NOLOG`. Otherwise the auditd daemon waits for the queue to have an empty spot before logging to disk. The risk is that while the daemon is waiting for network IO, an event is not being recorded to disk. Valid values are: `lossy` and `lossless`. Default: `lossy`.

dispatcher

The dispatcher is a program that is started by the audit daemon. It passes a copy of all audit events to stdin for that program. You should trust the application that you add to this line as it runs with root privileges.

name_format

Specifies how to insert computer node into the audit event stream. The name_ format parameter supports the following options:

- `none`
- `hostname`
- `fqd`
- `numeric`
- `user`

If set to `none`, no computer name is inserted into the audit event. If set to `hostname`, the name returned by the gethostname system call is inserted into the audit event. If set to `fqd` the hostname is resolved with DNS for a fully qualified domain name of the endpoint. If set to `numeric`, the hostname is resolved with the IP address of the endpoint. To use this option, test that 'hostname -i' or 'domainname -i' returns a numeric address. This option is not recommended for DHCP as it is possible to have different addresses over time for the same endpoint. `User` is an administrator defined string from the name option. Default: `None`.

name

Specifies the administrator defined string that identifies the endpoint if `user` is provided as the `name_format` option.

max_log_file

Specifies the maximum file size in megabytes. When this limit is reached, it triggers a configurable action. The value given must be numeric.

max_log_file_action

Specifies the action to take when the system has detected that the max file size limit has been reached. The max_log_file_action parameter supports the following options:

- `ignore`
- `syslog`
- `suspend`
- `rotate`
- `keep_logs`

If set to `ignore`, the audit daemon does nothing. If set to `syslog`, the audit daemon issues a warning to syslog. If set to `suspend`, the audit daemon stops writing records to the disk, but the daemon remains active. If set to `rotate`, the audit daemon rotates the logs with higher numbers being older than logs with lower numbers. The `keep_logs` option is similar to rotate except it does not use the `num_logs` setting to prevent audit logs from being overwritten. Default: `SYSLOG`.

action_mail_acct

Specifies a valid email address or alias. If the email address is not local to the endpoint, be sure that email is properly configured on the local computer and network. This option requires that `/usr/lib/sendmail` exists on the local computer. Default: `root`.

space_left

A numeric value in megabytes that instructs the audit daemon when to perform a configurable action when the local computer is starting to run low on disk space.

space_left_action

Specifies the action to take when the local system detects that it is starting to get low on disk space. The space_left_action parameter supports the following

options:

- `ignore`
- `syslog`
- `email`
- `exec`
- `suspend`
- `single`
- `halt`

If set to `ignore`, the audit daemon takes no action. If set to `syslog`, the audit daemon issues a warning to syslog. If set to `email`, the audit daemon sends a warning to the email account specified in `action_mail_acct` and sends the message to syslog. If set to `exec` the audit daemon executes a provided script. Note: you cannot pass parameters to the script. If set to `suspend`, the audit daemon stops writing records to the disk but the daemon remains active. If set to `single` the audit daemon puts the endpoint in single user mode. If set to `halt` the audit daemon shuts down the endpoint.

admin_space_left

A numeric value in megabytes that instructs the audit daemon when to perform a configurable action when the local computer is running low on disk space. This should be considered the last chance to do something before running out of disk space. The numeric value for this parameter should be lower than the number for `space_left`.

admin_space_left_action

Specifies the action to take when the system has detected that it is low on disk space. The admin_space_left_action parameter supports the following options:

- `ignore`
- `syslog`
- `email`
- `exec`
- `suspend`
- `single`
- `halt`

If set to `ignore`, the audit daemon takes no action. If set to `syslog`, the audit daemon issues a warning to syslog. If set to `email` the audit daemon sends a warning to the email account specified in `action_mail_acct` and sends the

message to syslog. If set to `syslog` the audit daemon issues a warning to syslog. If set to `exec`a provided script executes. You cannot pass parameters to the script. If set to `suspend`, the audit daemon stops writing records to the disk but the daemon is still active. If set to `single`, the audit daemon puts the endpoint in single user mode. If set to `halt`, the audit daemon shuts down the endpoint. Default: `SYSLOG`.

disk_full_action

Specifies the action to take when the system has detected that the partition to which log files are written has become full. The disk_full_action parameter supports the following options:

- `ignore`
- `syslog`
- `exec`
- `suspend`
- `single`
- `halt`

If set to `ignore`, the audit daemon issues a syslog message but no other action is taken. If set to `syslog` the audit daemon issues a warning to syslog. If set to `exec`a provided script executes. You cannot pass parameters to the script. If set to `suspend`, the audit daemon stops writing records to the disk but the daemon is still active. If set to `single`, the audit daemon puts the endpoint in single user mode. If set to `halt`, the audit daemon shuts down the endpoint. Default: `SYSLOG`.

disk_error_action

Specifies the action to take whenever there is an error detected when writing audit events to disk or rotating logs. The disk_error_action parameter supports the following options:

- `ignore`
- `syslog`
- `exec`
- `suspend`
- `single`
- `halt`

If set to `ignore`, the audit daemon issues up to five syslog messages before suppressing them and takes no other action. If set to `syslog` the audit daemon issues a warning to syslog. If set to `exec`, a provided script executes. You cannot pass parameters to the script. If set to `suspend`, the audit daemon stops writing

records to the disk but the daemon is still active. If set to `single`, the audit daemon puts the endpoint in single user mode. If set to `halt`, the audit daemon shuts down the endpoint. Default: `SYSLOG`.

tcp_listen_port

A numeric value in the range 1..65535 which, if specified, causes auditd to listen on the corresponding TCP port for audit records from remote systems. The audit daemon can be linked with tcp_wrappers. Use this option to control access with an entry in the `hosts.allow` and `deny` files.

tcp_listen_queue

A numeric value that indicates how many pending (requested but unaccepted) connections are allowed. Setting this too small causes connections to be rejected if too many hosts start at exactly the same time, such as after a power failure. Default: 5.

tcp_max_per_addr

A numeric value that indicates how many concurrent connections from one IP address are allowed. The maximum is 16. Setting this value large increases the possibility of a Denial of Service attack on the logging server. The default should be adequate in most cases unless a custom recovery script runs to forward unsent events. In this case, increase the number only large enough to let it in too. Default: 1.

use_libwrap

Specifies whether or not to use `tcp_wrappers` to discern connection attempts that are from allowed endpoints. Options are `yes` and `no`. Default: `yes`.

tcp_client_ports

A single numeric value or two values separated by a dash (without spaces). Indicates the client ports to use for incoming connections. If not specified, any port is allowed. Supported values are 1..65535. For example, to require the client use a privileged port, specify 1-1023 for this parameter. Set the `local_port` option in the `audisp-remote.conf` file. Verifying that clients send from a privileged port is a security feature to prevent log injection attacks by untrusted users.

tcp_client_max_idle

Specifies the number of seconds that a client can idle before auditd takes action. Use this option to close inactive connections if the client endpoint has a problem where it cannot shutdown the connection cleanly. Note that this is a global setting, and must be higher than any individual client **heartbeat_timeout** setting, preferably by a factor of two. Default: 0 (check disabled).

enable_krb5

If set to `yes`, Kerberos 5 is used for authentication and encryption. Default: `no`.

krb5_principal

Specifies the principal for the server. The server expects a key named in the following format: `auditd/hostname@EXAMPLE.COM` that is stored in `/etc/audit/audit.key` to authenticate itself. Hostname is the canonical name for the host of the server as returned by a DNS lookup of the IP address. Default: `auditd`.

krb5_key_file

Specifies the location of the key for the principal of the client. The key file must be owned by root and mode 0400. Default: `/etc/audit/audit.key`

CPU Killswitch parameters

cpuKillAfterNumViolations

The Client Recorder Extension continues to record system events until the maximum number of violations has been exceeded. Default: 1.

cpuKillEnabled

Toggles the CPU kill switch for the value provided by the cpuThreshold parameter. When set to false, the recorder does not shut down when threshold is exceeded. Default: True.

cpuThreshold

If the endpoint CPU use per processor exceeds this value over a one minute period, the Client Recorder Extension is disabled and audit rules are removed. Default: 25%.

cpuThresholdMinutes

Distributes the CPU use calculation over a multiple minute window. The maximum setting is 1440. Default: 1.

Database and filter locations parameters

dbLocation

The name and location of the Client Recorder Extension database. By default, the recorder database is located in the same directory as `config.json`. Default: `monitor.db`.

filterLocation

The name and location of the filters configuration. By default, this file is located in the same directory as `config.json`. Default: `filters.json`.

Logging parameters

logLevel

The level of logging to apply to Client Recorder Extension processes. Default: Information.

logMaxSize

The maximum size of the log file. Default: 10 MB.

logRotations

The number of times to roll log files when the maximum log file size is reached. Default: 3.

Maximum database size parameter

maxSizeMB

The maximum size of the Client Recorder Extension database (`monitor.db`). Default: 1024MB.

RAW Logging parameters

rawLogging

When set to true, audit rules are written to the raw logs and saved. This setting increases the audit log volume on the endpoint. When set to false, writing logs to disk is disabled. Use this setting for improved event throughput and lower CPU usage. Be sure that you do not have other, non-Tanium, processes that depend on reading raw audit logs. Default: True.

Miscellaneous Client Recorder Extension parameters

throttle

If set to True, information is logged to `/var/log/messages` and `recorder.log`. Processing is throttled to log events over a larger period of time to conserve resources and minimize potential disruption. Default: False.

onlyWatchFileRoot

Forces the recorder to watch the root folder. Integrity Monitor enters into an Integrity Monitor only file watch mode if this is not set. Default: True.

Windows registry entries

On Windows endpoints, Client Recorder Extension configuration data is contained in registry entries under `HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\Tanium\Tanium Client\Trace`.

Path

Specifies the working path for the Client Recorder Extension.

Type: REG_SZ

LogVerbosityLevel

Specifies the verbosity of the monitor log.

Type: REG_DWORD

Default Value = 1

Lower Bound = 0

Upper Bound = 100

LogBufferSizeInMessages

Specifies the number of messages in the Client Recorder Extension log file.

Type: REG_DWORD

Default Value = 1000

Lower Bound = 0

Upper Bound = 1000 * 1000

LogFileSize

Specifies the size of the Client Recorder Extension log file size in KB.

Type: REG_DWORD

LogFileSizeInBytes

Specifies the size of the Client Recorder Extension log file size in bytes.

Type: REG_DWORD

Default Value = settings["LogFileSize"]

Lower Bound = 0

Upper Bound = (1000 * 1024 * 1024)

LogFileSizeInMessages

Specifies the size of the Client Recorder Extension log file in the number of messages.

Type: REG_DWORD

Default Value = 0

Lower Bound = 0

Upper Bound = (1000 * 1000 * 1000)

IMToolsPath

Specifies the path to Tanium Integrity Monitor tools.

Type: REG_SZ

settings["path"] + \Tools\IM

TraceToolsPath

Specifies the path to Tanium Trace tools.

Type: REG_SZ

settings["path"] + \Tools\Trace

MapToolsPath

Specifies the path to Tanium Map tools.

Type: REG_SZ

settings["path"] + \Tools\Map

ForceIMOnlyMode

Specifies whether the Client Recorder Extension runs in Tanium Integrity Monitor only mode.

Type: REG_DWORD

Default Value = 0

Lower Bound = 0

Upper Bound = 1

MonitorDatabasePath

Specifies the path to the monitor database.

Type: REG_SZ

MonitorRegistry

Specifies whether to enable or disable registry monitoring.

Type: REG_DWORD

Default Value = 1

Lower Bound = 0

Upper Bound = 1

MonitorNetwork

Specifies whether to enable or disable network monitoring.

Type: REG_DWORD

Default Value = 1

Lower Bound = 0

Upper Bound = 1

MonitorFiles

Specifies whether to enable or disable file monitoring.

Type: REG_DWORD

Default Value = 1

Lower Bound = 0

Upper Bound = 1

MonitorDNS

Specifies whether to enable or disable DNS monitoring.

Type: REG_DWORD

Default Value = 1

Lower Bound = 0

Upper Bound = 1

MonitorImageLoad

Specifies whether to enable or disable image load monitoring.

Type: REG_DWORD

Default Value = 0

Lower Bound = 0

Upper Bound = 1

MonitorDays

Specifies the number of days of monitor data to hold in the database.

Type: REG_DWORD

Default Value = 90

Lower Bound = 0

Upper Bound = 65536

MaxStorageSizeMB

Specifies the maximum size of the `monitor.db` database. 0 is not limited.

Type: REG_DWORD

Default Value = 1024

Lower Bound = 0

Upper Bound = -1

MaxRuntimeStorageSizeMB

If the database exceeds this value while running it will set DisableTrace and exit. Defaults to 2 * MaxStorageSizeMB.

Type: REG_DWORD

Default Value = settings["MaxStorageSizeMB"] * 2

Lower Bound = 0

Upper Bound = -1

AbsoluteMaxStorageSizeMB

Specifies the maximum size of the `monitor.db` database. Default: The size of the `MaxStorageSizeMB`* 2.

Type: REG_DWORD

Default Value = settings["MaxStorageSizeMB"] * 2

Lower Bound = 0

Upper Bound = -1

CleanPercentOverLimit

Specifies the percentage by which to reduce the database past the `MaxStorageSizeMB`. Default 10.

Type: REG_DWORD

Default Value = 10

Lower Bound = 0

Upper Bound = 100

SysmonHistoryLimit

Specifies the amount of Sysmon history to load on startup in seconds.

Type: REG_DWORD

Default Value = 3600

Lower Bound = 0

Upper Bound = 86400

DatabaseCleanupIntervalMinutes

Specifies the amount of time between each database cleanup.

Type: REG_DWORD

Default Value = 30

Lower Bound = 1

Upper Bound = 2880

FilterDefinitionFile

Specifies the path to the filter definition JSON file.

Type: REG_SZ

DisableParentProcessFilter

Disables filtering the parent process when run as a DLL.

Type: REG_DWORD

Default Value = 0

Lower Bound = 0

Upper Bound = 1

BatchWriteDelayMS

Specifies the delay, in milliseconds, between each batch save operation.

Type: REG_DWORD

Default Value = 1000

Lower Bound = 0

Upper Bound = 5000

DisableSetDebug

Disables setting the `SE_DEBUG_PRIVILEGE` flag. Setting this to 1 can cause the process list to be less comprehensive.

Type: REG_DWORD

Default Value = 0

Lower Bound = 0

Upper Bound = 1

ResetIntervalMinutes

Specifies the amount of time the DLL runs before being reset in minutes.

Type: REG_DWORD

Default Value = 240

Lower Bound = 0

Upper Bound = 10080

NewEventLimit

Specifies the maximum number of events to be created. A lower value for this can increase CPU usage but lowers memory utilization.

Type: REG_DWORD

Default Value = 2000

Lower Bound = 0

Upper Bound = 100000

MaxCleanTries

Specifies the number of times to attempt to clean the database. If it cannot be cleaned in this number of attempts, the database is archived.

Type: REG_DWORD

Default Value = 3

Lower Bound = 0

Upper Bound = 10

DatabaseArchiveLimit

Specifies the maximum number of database archives to keep. Erases the oldest if it is above this limit.

Type: REG_DWORD

Default Value = 0

Lower Bound = 0

Upper Bound = 100

DisableTrace

Disables the Client Recorder Extension.

Type: REG_DWORD

Default Value = 0

Lower Bound = 0

Upper Bound = 1

WMICTimeout

Specifies the timeout for the `WMIC` command in seconds. 0 disables the command.

Type: REG_DWORD

Default Value = 0

Lower Bound = 30

Upper Bound = 300

DatabaseErrorWaitMinutes

Specifies the amount of time to wait prevent high resource usage when the `monitor.db` database creation fails.

Type: REG_DWORD

Default Value = 5

Lower Bound = 0

Upper Bound = 60

PendingEventCap

Controls CPU/memory usage under load. If the number of events currently being processed exceeds this value, all incoming events are dropped until the number of events is fewer than this threshold.

Type: REG_DWORD

Default Value = 5000

Lower Bound = 0

Upper Bound = -1

DropRegCreateKey

Disables logging of `RegCreateKey`.

Type: REG_DWORD

Default Value = 1

Lower Bound = 0

Upper Bound = 1

WatchlistFile

Specifies the path to the Tanium Integrity Monitor watchlist file.

Type: REG_SZ

Default Value = tanium client path \watchlist.json

ProcessStateFile

Specifies the path to the process state file.

Type: REG_SZ

Default Value = tanium client path \Trace\process_state.bin

ImageHashStateFile

Specifies the path to the image state file.

Type: REG_SZ

Default Value = tanium client path \Trace\image_state.bin

ProcessExpirationSeconds

Specifies the number of seconds to hold process information in memory after closed.

Type: REG_DWORD

Default Value = 180

Lower Bound = 0

Upper Bound = -1

SignalsJSONFile

Path to Signals JSON definition.

Type: REG_SZ

Default Value = tanium client path \Trace\signals.json

SignalsBinFile

Specifies the path to encoded Signals data.

Type: REG_SZ

Default Value = tanium client path \Trace\signals.bin

DisableSignals

Disables Tanium Signals processing.

Type: REG_DWORD

Default Value = 0

Lower Bound = 0

Upper Bound = 1

DisableSignalsTrie

Disables Signals processing for contains/begins with/ends with.

Type: REG_DWORD

Default Value = 0

Lower Bound = 0

Upper Bound = 1

MaxFileHandlesPerProcess

Specifies the maximum number of file handles to hold for a process.

Type: REG_DWORD

Default Value = 500

Lower Bound = 0

Upper Bound = -1

SysmonLowerToleranceMS

Specifies the lower tolerance in milliseconds used when mapping sysmon events to process events.

Type: REG_DWORD

Default Value = 200

Lower Bound = 0

Upper Bound = -1

SysmonUpperToleranceMS

Specifies the upper tolerance in milliseconds used when mapping sysmon events to process events.

Type: REG_DWORD

Default Value = 200

Lower Bound = 0

Upper Bound = -1

MappingLowerToleranceMS

Specifies the lower tolerance in milliseconds used when mapping events to process events.

Type: REG_DWORD

Default Value = 200

Lower Bound = 0

Upper Bound = -1

MappingUpperToleranceMS

Specifies the upper tolerance in milliseconds used when mapping events to process events.

Type: REG_DWORD

Default Value = 200

Lower Bound = 0

Upper Bound = -1

DNSEventIDList

A comma separated list of event IDs for monitored DNS client events. Must be in set notation {3008,3013,3018,3020}.

Type: REG_DWORD

Default Value = "3008,3013,3018,3020"

ForceSysmon

Forces the use of Sysmon even when alternatives are available.

Type: REG_DWORD

Default Value = 0

Lower Bound = 0

Upper Bound = 1

ForceRecorderDriver

Forces the use of the Client Recorder Extension driver.

Type: REG_DWORD

Default Value = 0

Lower Bound = 0

Upper Bound = 1

UseAuditCommandLine

Specifies to use the security audit log for command line.

Type: REG_DWORD

Default Value = 0

Lower Bound = 0

Upper Bound = 1

VerboseSignals

Enables verbose debugging for signals.

Type: REG_DWORD

Default Value = 0

Lower Bound = 0

Upper Bound = 1

CommandLineMappingRetry

Specifies the number of save batches a command line attempts.

Type: REG_DWORD

Default Value = 10

Lower Bound = 0

Upper Bound = 10

SysmonCheckIntervalMinutes

Specifies the interval that Sysmon status is checked in minutes.

Type: REG_DWORD

Default Value = 5

Lower Bound = 0

Upper Bound = 300

ETWCustomBuffers

Enables custom ETW buffer sizes.

Type: REG_DWORD

Default Value = 1

Lower Bound = 0

Upper Bound = 1

ETWBufferSize

Specifies the amount of memory allocated for each event tracing session buffer, in kilobytes. The maximum buffer size is 1 MB.

Type: REG_DWORD

Default Value = 1024

Lower Bound = 0

Upper Bound = 1024

ETWMinBuffer

Specifies the minimum number of buffers allocated for the buffer pool for the event tracing session.

Type: REG_DWORD

Default Value = settings["ETWMaxBuffer"]

Lower Bound = processor count * 2

Upper Bound = 128

ETWMaxBuffer

Specifies the maximum number of buffers allocated for the buffer pool for the event tracing session.

Type: REG_DWORD

Default Value = minBufferCount + 20

Lower Bound = processor count * 2

Upper Bound = 128

RecordSignalDefinitions

Enables storing compiled signal definitions in the `monitor.db` database.

Type: REG_DWORD

Default Value = 1

Lower Bound = 0

Upper Bound = 1

RecordSignalMatches

Specifies the number of matched signals to store in the `monitor.db` database.

Type: REG_DWORD

Default Value = 5

Lower Bound = 0

Upper Bound = -1

DisableImageLoadHashing

Type: REG_DWORD

Default Value = 0

Lower Bound = 0

Upper Bound = 1

FileExtraFlagsFilter

Filters file write events that match this mask in the `ExtraFlags` field.

Type: REG_DWORD

Default Value = 0

Lower Bound = 0

Upper Bound = -1

ConfirmFileWrites

Verifies that the file modification time changes on file writes.

Type: REG_DWORD

Default Value = 0

Lower Bound = 0

Upper Bound = 1

SQLiteChunkSizeMB

Specifies the size of chunks in MB allocated from the file system when the `monitor.db` database size increases.

Type: REG_DWORD

Default Value = 100

Lower Bound = 0

Upper Bound = -1

EventQueueExpirationSeconds

Specifies the number of seconds to wait for a process event before purging pending events.

Type: REG_DWORD

Default Value = 120

Lower Bound = 1

Upper Bound = -1

Managing configurations across modules

The Client Recorder Extension reads `<product name>_recorder.json` and `<product_name>_filters.json` in the `<Tanium Client Dir>/Tools/Trace` directory. Tanium Trace currently uses these configuration files without `<product name>_`. These configuration files are read in by the Client Recorder Extension, and it merges that information together to satisfy respective product goals.

Field	Strategy	Default Value
maxSizeMB	max value	1024
logRotations	max value	3
logMaxSize	min value	10 MB
rawLogging (Linux only)	any is false, then false	Is not set if not set in a configuration file.
cpuThreshold	min value	0.25
cpuKillEnabled (Linux and Mac only)	if any is false, kill switch is disabled	True

Troubleshooting the Client Recorder Extension

Identify Linux endpoints missing auditd

If Linux endpoint events are not being recorded, they might be missing the audit daemon and audispd. Ideally, the audit daemon is installed and configured before installing the Trace module, but it is possible for endpoints to come online at a later time.

1. (Optional) Create the auditd package.
   You can either create a general installation package and put the logic in the scripts or you can have a simple script and put the logic in the Tanium query. See Tanium Core Platform User Guide: Creating and managing packages.

   > **Tip:** Create saved actions that periodically check for and deploy this package in the future.

2. Ask the question: `Get Installed Application Exists[audit] from all machines with Is Linux containing "true"`.
3. Use your preferred method to deploy the appropriate auditd package to the identified endpoints.

   > **IMPORTANT:** If you need to distribute the package to a large number of endpoints, spread the changes out over time to avoid a negative impact on the network.

Re-create the endpoint database

If the database is identified as corrupt, you might need to clear the endpoint database.

1. Use a question to target the affected endpoints.
   For example, ask `Get Trace Invalid File Operations from all machines`.
2. Drill down to the endpoints that return `true`.
3. Deploy the **Trace - Recreate Database [OS]** or **Threat Response - Recreate Database [OS]** package as an action.

For more information, see the Tanium Core Platform User Guide: Managing and creating Packages or the Tanium Interact User Guide: Using Deploy Action.