Tanium™ Client Recorder Extension User Guide

Version 2.1.0

January 17, 2020

Table of contents

Client Recorder Extension overview

The Client Recorder Extension is a feature common to the Tanium Integrity Monitor, Tanium Map, and Tanium Threat Response solution modules. It continuously saves event data 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 returns event information based on a subscription that a module provides. Subscriptions can save event information in a number of ways, for example in JSON format or in a database. Modules 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 create subscriptions to capture specific 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 data 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.

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 database and/or journal feeds. Kernel events are gathered from Windows tools. On Windows endpoints, the Tanium Driver is recommended to provide 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.

The Client Recorder Extension for Linux requires an application that is started by auditd. In the Client Recorder Extension for Linux, the Tanium Auditpipe is the application that auditd starts. When a Client Recorder Extension configuration is provided, auditd is restarted, and in turn starts the Tanium Auditpipe. When a configuration is removed, auditd also restarts. For this reason you can see auditd

starting and stopping when Client Recorder Extension configurations are added or removed.

The `log_format` parameter in `auditd.conf` has been deprecated in versions of auditd 2.5.2 and later. The `log_format` parameter has two settings:

- `RAW`
- `NOLOG`

If the auditd.conf contains `log_format = NOLOG` on these versions of auditd, the audispd process does not start. To disable RAW logging on these versions, change the following parameters in auditd.conf:

- `write_logs = NO`
- `log_format = RAW`

When the Client Recorder Extension starts on an endpoint that has auditd version 2.5.2 or later, it changes `log_format = NOLOG` to `log_format = RAW`. If the `write_log` parameter is detected in `auditd.conf`, the value of the `log_format` parameter is not changed. If the `write_log` parameter is not detected, it is added to `auditd.conf` corresponding to `RAW` or `NOLOG`.

For example:

- If `log_format = NOLOG` and `write_logs` is not set, the Client Recorder Extension sets `log_format = RAW` and `write_logs = NO`
- If `log_format = NOLOG` and `write_logs = YES`, the Client Recorder Extension sets `log_format = RAW` and does not make changes to the value of the `write_logs` parameter.

When the Client Recorder Extension starts on an endpoint that has a version of auditd earlier than 2.5.2, the `write_logs` parameter is removed.

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/`.

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

Sources of Client Recorder Extension data on Mac

On Mac endpoints, the Client Recorder Extension collects data from:

- The OpenBSM auditing system that is installed in all Mac releases from 10.8 to current.
- FSevents, which enables applications to register for notifications of changes to a directory tree.
- Kextd, which provides information about kernel extensions.

The Client Recorder Extension connects to a clone of `/dev/auditpipe` to record events. 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 Client Recorder Extension then clones `/dev/auditpipe` and configures the copy to use the `tan` audit class. When the Client Recorder Extension 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 Client Recorder Extension from writing the audited events to the audit.log of the endpoint and allows the Client Recorder Extension to be very selective about which system calls are monitored.

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.

Getting started

1. Install the Client Recorder Extension. For more information, see Installing the Client Recorder Extension on page 17.
2. Configure endpoint settings. For more information, see Configuring recorded events on page 23.

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	7.2.314.3550 or later. 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 best results, the following Tanium Client versions are suggested: • 6.0.314.1540 (Windows) • 7.2.314.3211 (Linux, MacOS, Windows) • 7.2.314.3476 (Linux, MacOS, Windows) • 7.2.314.3518 (Linux, MacOS, Windows) 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	• 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 4 GB RAM is recommended on each endpoint device. By default, the endpoint database for Threat Response is between 256 MB and 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 service needs to be restarted.

For full-functionality a minimum of two CPUs per endpoint is recommended.

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

For Mac endpoints, macOS 10.11 or later is required.

The Recorder Client Extension is not supported on AIX or Solaris endpoints.

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

Target Device	Process
Endpoint computers (Windows)	*<Installation Location>*\sysmon.exe
	<Tanium Client>\extensions\TaniumRecorder.dll
	<Tanium Client>\extensions\TaniumRecorder.dll.sig
	<Tanium Client>\extensions\recorder\proc.bin
	<Tanium Client>\extensions\recorder\recorder.db
	<Tanium Client>\extensions\recorder\recorder.db-shm
	<Tanium Client>\extensions\recorder\recorder.db-wal
	<Tanium Client>\extensions\recorder\<sample_database>.json

Target Device	Process
Endpoint computers (Linux)	*<Tanium Client>*/extensions/libTaniumRecorder.so
	<Tanium Client> /extensions/libTaniumRecorder.so.sig
	<Tanium Client>/extensions/recorder/proc.bin
	<Tanium Client>/extensions/recorder/recorder.db
	<Tanium Client> /extensions/recorder/recorder.db-shm
	<Tanium Client> /extensions/recorder/recorder.db-wal
	<Tanium Client>/extensions/recorder/<sample_ database>.json
	<Tanium Client> /extensions/recorder/recorder.auditpipe
	/etc/audisp/plugins.d/tanium.conf

Target Device	Process
Endpoint computers (Mac)	*<Tanium Client>* /extensions/libTaniumRecorder.dylib
	<Tanium Client> /extensions/libTaniumRecorder.dylib.sig
	<Tanium Client>/extensions/recorder/proc.bin
	<Tanium Client>/extensions/recorder/recorder.db
	<Tanium Client> /extensions/recorder/recorder.db-shm
	<Tanium Client> /extensions/recorder/recorder.db-wal
	<Tanium Client>/extensions/recorder/<sample_database>.json
	<Tanium Client> /extensions/recorder/recorder.auditpipe
	/etc/audisp/plugins.d/tanium.conf

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/extensions/libTaniumRecorder.so` (Linux)
`/opt/Tanium/TaniumClient/extensions/libTaniumRecorder.dylib` (Mac)
`C:\Program Files(x86)\Tanium\Tanium Client\extensions\TaniumRecorder.dll` (Windows)

The Client Recorder Extension process.

`/opt/Tanium/TaniumClient/extensions/libTaniumRecorder.so.sig` (Linux)
`/opt/Tanium/TaniumClient/extensions/libTaniumRecorder.dylib.sig` (Mac)
`C:\Program Files(x86)\Tanium\Tanium Client\extensions\TaniumRecorder.dll.sig` (Windows)

A signature file that you can use to verify that the contents of the .so, .dylib, or .dll file is authentic and have not been tampered with.

`/opt/Tanium/TaniumClient/extensions/recorder/proc.bin` (Linux)
`/opt/Tanium/TaniumClient/extensions/recorder/proc.bin` (Mac)
`C:\Program Files(x86)\Tanium\Tanium Client\extensions\recorder\proc.bin` (Windows)

Captures an enumeration of processes that were running when Client Recorder Extension is stopped the so a delta can be captured when the Client Recorder Extension is started. For example, the last known signal states are recorded.

`/opt/Tanium/TaniumClient/extensions/recorder/recorder.db` (Linux)
`/opt/Tanium/TaniumClient/extensions/recorder/recorder.db`

(Mac)
C:\Program Files(x86)\Tanium\Tanium Client\extensions\recorder\recorder.db (Windows)

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

/opt/Tanium/TaniumClient/extensions/recorder/recorder.db-shm (Linux)
/opt/Tanium/TaniumClient/extensions/recorder/recorder.db-shm (Mac)
C:\Program Files(x86)\Tanium\Tanium Client\extensions\recorder\recorder.db-shm (Windows)

A shared memory file. Database connections that share the same db file must update the same memory location to prevent conflicts.

/opt/Tanium/TaniumClient/extensions/recorder/recorder.db-wal (Linux)
/opt/Tanium/TaniumClient/extensions/recorder/recorder.db-wal (Mac)
C:\Program Files(x86)\Tanium\Tanium Client\extensions\recorder\recorder.db-wal (Windows)

A write journal that is useful for commits and database rollback purposes.

/opt/Tanium/TaniumClient/extensions/recorder/<*sample_database*>.json (Linux)
/opt/Tanium/TaniumClient/extensions/recorder/<*sample_database*>.json (Mac)
C:\Program Files(x86)\Tanium\Tanium Client\extensions\recorder\<*sample_database*>.json (Windows)

A sample database.

/opt/Tanium/TaniumClient/extensions/recorder/recorder.auditpipe (Linux)
/opt/Tanium/TaniumClient/extensions/recorder/recorder.auditpipe (Mac)

An auditpipe that receives forwarded events from audispd that is created by `/opt/Tanium/TaniumClient/TaniumAuditPipe`. For systems that have SE Linux, a Tanium Client and a Tanium Recorder policy are installed.

`/etc/audisp/plugins.d/tanium.conf` (Linux)

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. If `augenrules` exists on the system, audit rules are also generated to `/etc/audit/rules.d/tanium.rules`.

Tanium Recorder Driver (Windows)

The driver is installed to the following location by default:

`%windir%\system32\drivers\TaniumRecorderDrv.sys`

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.

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

A configuration file controls the configuration of the audit event dispatcher process. The Client Recorder Extension modifies the q_depth setting to 32768. q_depth is the only setting that is configured by default.

`/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.

Starting and stopping the Client Recorder Extension

You might need to manually start or stop the Client Recorder Extension. For example, when troubleshooting you must resolve the underlying issue first and then manually restart the Client Recorder Extension. Or, if you find that the Client Recorder Extension

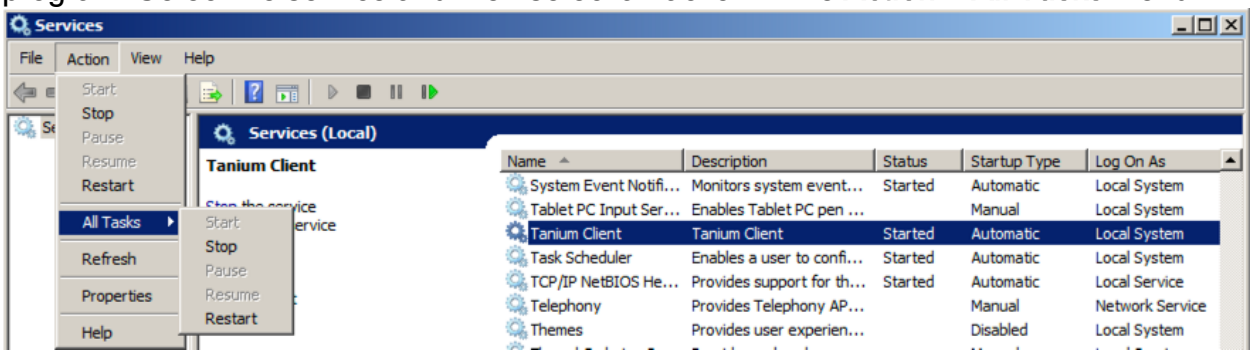
is using more system resources than expected, you can stop the Client Recorder Extension and troubleshoot the issue with the risk of additional resource consumption.

The Client Recorder extension starts when a configuration is deployed. Without a configuration, the Client Recorder Extension is idle. When a configuration is provided the Client Recorder Extension is granted permission to interface with operating system processes.

In the event of a troubleshooting situation, you can stop or start the Client Recorder Extension by stopping and starting the Tanium Client. Since the Client Recorder Extension starts and stops depending on the availability of a configuration, this is not a common necessity.

Windows endpoints

You can stop, start, or restart the **Tanium Client** service through the Windows **Services** program. Select the service and then select an action in the **Action > All Tasks** menu.



Mac endpoints

Use the **launchctl** command to manage the Tanium Client service.

To start the Tanium Client service:

```
sudo launchctl load /Library/LaunchDaemons/com.tanium.taniumclient.plist
```

To stop the Tanium Client service:

```
sudo launchctl unload /Library/LaunchDaemons/com.tanium.taniumclient.plist
```

To remove the Tanium Client from the launch list:

```
sudo launchctl remove com.tanium.taniumclient
```

Linux Endpoints

Linux service commands vary according to Linux distribution. This documentation provides examples but is not a reference for each Linux distribution. If you are not already familiar with installing and managing services on your target Linux distribution, please review the documentation for the particular Linux operating system before you begin.

Linux Distribution	Example Commands
Amazon Linux	`service TaniumClient start` `service TaniumClient stop`
Debian	`service taniumclient start` `service taniumclient stop`
Oracle Enterprise Linux	`systemctl start taniumclient` (Version 7) `systemctl stop taniumclient` (Version 7) `service TaniumClient start` (Version 5, 6) `service TaniumClient stop` (Version 5, 6)
Red Hat / CentOS	`systemctl start taniumclient` (Version 7) `systemctl stop taniumclient` (Version 7) `service TaniumClient start` (Version 5, 6) `service TaniumClient stop` (Version 5, 6)
SUSE / OpenSUSE	`service taniumclient start` `service taniumclient stop`
Ubuntu	`systemctl start taniumclient` (Version 16) `systemctl stop taniumclient` (Version 16) `service taniumclient start` (Versions 14, 10) `service taniumclient stop` (Version 14, 10)

(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 recorded events

The Client Recorder extension collects data from different sources depending on the operating system of the endpoint. For more information on the sources of event data for each operating system, see Sources of Client Recorder extension data. Each data source provides event information into a queue in a generic format. The Client Recorder extension then converts the generic events into a format that can be written to a database or other data stream. If necessary, the Tanium Client can manage resources.

Configure the events that the Client Recorder extension records by defining a subscription. There are several types of subscriptions, such as subscriptions for databases and journals. A subscription has a name, a target domain and an array of streams. The stream configures the types of events that you want to record. For example, a database subscription has an array of `event_filters` that define which types of events the Client Recorder extension writes to a database. Each `event_filter` has an array of `terms`. The `terms` are joined by a logical AND; for example, `process and process.path contains not 'C:\Program Files'` to include all process paths with the exception of those that are in `C:\Program Files`. All `event_filters` are joined by a logical OR; for example, `process and (process.path contains not 'C:\Program Files' and process.path contains not 'C:\Windows' and process.path contains not 'C:\Users')`.

The following match criteria are supported by the Client Recorder extension:

`process.path`

`process.cmd`

`process.hash`

`process.user_name`

`process.user_group`

`process.parent_path`

`process.parent_cmd`

`process.ancestry_path`

`process.ancestry_cmd`

`process.ancestry_hash`

`process.ancestry_user_name`

`process.ancestry_user_group`

```
registry.name

registry.value

file.path

file.operation

network.addr

network.port

dns.query

image.path

image.hash

security_event.id
```

When the conditions that you have configured the Client Recorder extension with are met, storable events are committed to the database (for a database subscription) or other data stream. The output format for a database subscription is a SQLlite database named `recorder.db`.

When you update a recorder subscription and register it with the Client Recorder Extension, the recorded events are updated without restarting the Client Recorder Extension. For more information, see Client Recorder Extension commands.

Modules that use the Client Recorder Extension provide a default configuration that is registered with the Client Recorder Extension when you install them. 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 settings can have serious, and sometimes irrevocable consequences.

Global configurations

The `auditctl --backlog_wait_time 0` is added to all Linux configurations. On newer kernels (> 3.14) the `backlog_wait_time` setting can cause a kernel panic. Setting this value to zero by default is a preventative measure to mitigate unstable kernel performance.

Client Recorder Extension configuration settings

CX.recorder.AudispdMaxRestarts

Minimum audispd max restarts. Default: 10.

CX.recorder.AudispdMaxRestartsWrite

Enables the recorder to use the value set in AudispdMaxRestarts to update audispd.conf. Default: 0.

CX.recorder.AudispdOverflowAction

Specifies the audispd overflow action.

CX.recorder.AudispdOverflowActionWrite

Allow the recorder to use the value set in AudispdOverflowAction to update audispd.conf. Default: 0.

CX.recorder.AudispdPriorityBoost

Minimum audispd priority_boost. Default: 4.

CX.recorder.AudispdPriorityBoostWrite

Enables the recorder to use the value set in AudispdPriorityBoost to update audispd.conf. Default: 0

CX.recorder.AudispdQueueDepth

Minimum audispd q_depth. Default: 32768.

CX.recorder.AudispdQueueDepthWrite

Enables the recorder to use the value set in AudispdQueueDepth to update audispd.conf. Default: 1.

CX.recorder.AuditdDisableKernelPanicProtection

Enables the recorder to run when auditd is in panic failure mode. Default: 0.

CX.recorder.AuditDMaxWatchedPaths

Maximum number of individual paths watched before giving up and watching root. Default: 50.

CX.recorder.AuditdRawLogging

Enable/Disable auditd raw logging. Default: 0

CX.recorder.AuditdRawLoggingWrite

Enables the recorder to use the value set in AuditdRawLogging to update auditd.conf. Default: 0.

CX.recorder.AuditdRulesBufferSize

Buffer size set in audit rules. Default: 8192.

CX.recorder.AuditdRulesDisableWatchSpecificPaths

Watch specific paths driven by subscription in auditd. Default: 0.

CX.recorder.ConfigRefreshConfigInterval

Specifies how often to forceably refresh operating system config state. Default: 60.

CX.recorder.ConfigRefreshInterval

Specifies how often to refresh configuration updates. Default: 60.

CX.recorder.DatabaseMaxSizeMB

Specifies the max size of the database before cleaning in MB. Default: 924.

CX.recorder.EmitFileWriteIntervalSeconds

Specifies the interval (in seconds) between emitting file write events for the same file handle. Default: 30.

CX.recorder.EnableEtw

Enables Windows ETW events. Default: 1

CX.recorder.EnableEtwDns

Enables Windows ETW for DNS Events. Default: 1.

CX.recorder.EnableEtwFile

Enables Windows ETW for File Events. Default: 1

CX.recorder.EnableEtwNetwork

Enables Windows ETW for Network Events. Default: 1.

CX.recorder.EnableEtwProcess

Enables Windows ETW for Process Events. Default: 1

CX.recorder.EnableEtwRegistry

Enables Windows ETW for Registry Events. Default: 1.

CX.recorder.EnableEtwSysmon

Enables Windows ETW for Sysmon Events. Default: 0.

CX.recorder.EnableEtwTaniumDriver

Enables Windows ETW for Tanium Driver Events. Default: 1.

CX.recorder.EnableEvt

Enables Windows Eventlog events. Default: 1

CX.recorder.EnableKEtw

Enables Windows Kernel ETW events. Default: 1

CX.recorder.EnableLibraryHashing

Enables hashing of library load events. Default: 0.

CX.recorder.EnableMacBSM

Enables BSM on macOS. Default: 1.

CX.recorder.EnableMacFSEvents

Enables FSEvents on macOS. Default: 1.

CX.recorder.EnableMacKext

Enables kext load events on macOS. Default: 1.

CX.recorder.EnableWinDLLSigCheck

Enables checking all loaded DLLs for signatures on Windows. Default: 0.

CX.recorder.EnableWinEXESigCheck

Enables checking executables for signatures on Windows. Default: 1.

CX.recorder.EnableWinSYSSigCheck

Enables checking loaded drivers for signatures on Windows. Default: 1.

CX.recorder.EtwMatchingToleranceMs

Specifies the milliseconds of tolerance allowed in ETW when matching events to a process. Default: 100.

CX.recorder.EventAssemblerTimeoutMs

Specifies the number of MS assembler will wait to complete a process context before forwarding. Default: 10000.

Cx.recorder.FileInfoProviderDomain

Specifies the domain of FileInfoProvider used to hash executables. Default: recorder.

CX.recorder.FileInfoProviderName

Specifies the name of the FileInfoProvider that is used to hash executables. Default: file_info

CX.recorder.HashCacheSize

Specifies the size of the hash cache in Processor hash provider. Default: 50.

CX.recorder.JournaldRawLogging

Enable/Disable systemd journald auditd socket raw logging. Default: 0.

CX.recorder.JournaldRawLoggingWrite

Enables the recorder to use the value set in JournaldRawLogging to call systemctl. Default: 0.

CX.recorder.PathReassemblyTimeoutSeconds

Specifies the number of seconds the event converter will wait to assemble a path before deleting. Default: 300.

CX.recorder.PersistenceQueueDrainIntervalMs

Specifies how often (in milliseconds) to check the queue for processed events. Default: 100.

CX.recorder.PersistenceWriteProcessIntervalSecs

How often to periodicially write the proc.bin file to disk. Default: 600.

CX.recorder.ProcessorCleanupIntervalSecs

Specifies how often to perform cleanups in the processor. Default: 60.

CX.recorder.ProcessorMaxNonProcessItemsPerDrain

Specifies how many non-process items to service per drain. Default: 60.

CX.recorder.ProcessorMaxProcessItemsPerDrain

Specifies how many process items to service per drain. Default: 1024.

CX.recorder.ProcessorPruneAdjustmentSecs

Specifies any additional tolerance required before the processing stage prunes caches. Default: 60.

CX.recorder.ProcessorQueueDrainIntervalMs

Specifies how often (in milliseconds) to check the queue for parsed events. Default: 100.

CX.recorder.SnapshotExtraSizeRequiredMB

Specifies the required extra free space (in MB) to be available to perform a snapshot. Default: 1024.

CX.recorder.SnapshotWriteIntervalMs

Specifies the interval that snapshots will write pages to the destination database. Default: 100.

CX.recorder.SnapshotWriteSizeMB

Specifies the size of each write, in megabytes, that the snapshot will write each SnapshotWriteIntervalMs interval. Default: 24.

CX.recorder.Stage1WorkQueueInterval

Specifies how often to pull items from the stage one worker queues in milliseconds. Default: 10.

CX.recorder.StatusUpdateInterval

Specifies how often to update status information. Default: 60.

CX.recorder.VacuumIntervalDays

The number of days between each database vacuum. 0 is disabled. Default: 7.

Client Recorder Extension commands

When you have created a Client Recorder Extension configuration, you can control the Client Recorder Extension by issuing commands.

recorder.register-subscription

Adds a subscription to the Client Recorder Extension. To update or change a subscription, issue this command with updated configuration information and use the same name and domain. The name and domain are configured by using the FileInfoProviderName and FileInfoProviderDomain configuration settings.

recorder.get-subscription

Returns the subscription the product registered.

recorder.remove-subscription

Removes subscriptions added by recorder.register-subscription, on the last subscription removal uninstall is called and removes all audit rules.

recorder.query

Constructs a SQL command and run against the recorder database.

recorder.snapshot

Backs up the recorder database to a directory. This is how THR backs up the recorder database.

recorder.uninstall

Ensures that the event sources are disabled, audit rules are removed properly, and all subscriptions are removed. The effect of returning the system back to initial install. The exception when the auditing system is in immutable mode, on system reboot, it returns to initial state.

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.