Tanium™ Client Recorder Extension User Guide

Version 2.9.0

March 20, 2023

The information in this document is subject to change without notice. Further, the information provided in this document is provided "as is" and is believed to be accurate, but is presented without any warranty of any kind, express or implied, except as provided in Tanium's customer sales terms and conditions. Unless so otherwise provided, Tanium assumes no liability whatsoever, and in no event shall Tanium or its suppliers be liable for any indirect, special, consequential, or incidental damages, including without limitation, lost profits or loss or damage to data arising out of the use or inability to use this document, even if Tanium Inc. has been advised of the possibility of such damages.

Any IP addresses used in this document are not intended to be actual addresses. Any examples, command display output, network topology diagrams, and other figures included in this document are shown for illustrative purposes only. Any use of actual IP addresses in illustrative content is unintentional and coincidental.

Please visit https://docs.tanium.com for the most current Tanium product documentation.

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.

Tanium is committed to the highest accessibility standards for our products. To date, Tanium has focused on compliance with U.S. Federal regulations - specifically Section 508 of the Rehabilitation Act of 1998. Tanium has conducted 3rd party accessibility assessments over the course of product development for many years and has most recently completed certification against the WCAG 2.1 / VPAT 2.3 standards for all major product modules in summer 2021. In the recent testing the Tanium Console UI achieved supports or partially supports for all applicable WCAG 2.1 criteria. Tanium can make available any VPAT reports on a module-by-module basis as part of a larger solution planning process for any customer or prospect.

As new products and features are continuously delivered, Tanium will conduct testing to identify potential gaps in compliance with accessibility guidelines. Tanium is committed to making best efforts to address any gaps quickly, as is feasible, given the severity of the issue and scope of the changes. These objectives are factored into the ongoing delivery schedule of features and releases with our existing resources.

Tanium welcomes customer input on making solutions accessible based on your Tanium modules and assistive technology requirements. Accessibility requirements are important to the Tanium customer community and we are committed to prioritizing these compliance efforts as part of our overall product roadmap. Tanium maintains transparency on our progress and milestones and welcomes any further questions or discussion around this work. Contact your sales representative, email Tanium Support at support@tanium.com, or email accessibility@tanium.com to make further inquiries.

Table of contents

Client Recorder Extension overview

The Client Recorder Extension is a feature common to the Tanium Enforce, 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
- file read events
- registry changes
- network connections
- driver and library loads
- user authentication
- http headers
- operating system state changes (login/logout)

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. The Client Recorder Extension does not collect any information regarding the output redirection. As the Client Recorder Extension works at the process level, the Client Recorder Extension does not include shell builtins, such as '|', "<", ">", or ">>" as part of the process for a command line and treats commands separated by these into multiple processes and handles each of the processes individually.

For example, the command `ps aux | grep root` is interpreted as two commands:

```
ps aux
```

and

`grep root`

Similarly, the command `svnadmin dump c:\path\to\myrepo | 7z a -si svndump.7z` is interpreted as two commands:

`svnadmin dump c:\path\to\myrepo`

and

`7z a -si svndump.7z`

Each of these processes share the same parent process.

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.

IMAGE LOAD AND REGISTRY

[Windows only] Changes to the registry, such as the creation, renaming, or alteration of registry keys and values. Includes the associated process and user context. Image Loads record the process GUID and ensuring a correct PID match.

NETWORK

Network connection events, including the associated process and user context. Events are recorded for all inbound and outbound TCP connections. The Client Recorder Extension tracks the `bind` system call for network events. For Linux endpoints that use auditd as an event source, all outgoing connections do not display local address and port information. For Linux endpoints that use eBPF as an event source, outgoing connections display local address and port information.

FILE

File system events, such as files written to directory locations on the endpoint. The associated process and user context are included. When a process is executed, the original login user - if it exists - is recorded. For example if a user creates an SSH connection to an endpoint, and then becomes some other user, the user name is the user the process executed as but the login user is the original user who logged in. This can be useful for filtering events for a certain user or user id.

Examples of file system events include a malware file copied to a location that Windows Update uses, or content changes made to a file. The Client Recorder Extension nomalizes file paths to provide support for symlinks.

File read events are useful for determining if someone other than a certain user has tried to read a file. Custom auditd rules are required to enable read events. File open read will behave similarly to File Open Write. To support file read auditing, users with administrative permissions can upload a new segment of auditd rules through the Client Recorder Extension for Linux. Custom audit rules allow the administrator to manage their own rules into auditd using the following commands:

- `recorder.register-user-defined-audit-rules`
- `recorder.get-user-defined-audit-rules`
- `recorder.remove-user-defined-audit-rules`

On successful addition of user defined rules, Client Extensions - Status reports:

```
{
"domain": "recorder",
"name": "has_user_defined_audit_rules",
"value": "true"
}
```

On failure to refresh rules with custom rules, the custom rules are removed. Only rules starting with -a or -A are accepted. Rules with dir= or path= check the dir/path if it exists on the local system before they are added. If custom audit rules produce an error when loading rules, the custom rules are removed and a health check presented.

You can view file read events in the recorder database, through integrations to Stream, in Detect signals, and in journal subscriptions.

SECURITY

Security events such as authentication, privilege escalation, and more. This event type includes logon events.

DNS

[Windows 8.1 or later and Linux endpoints that use eBPF as an event source] Request information, including the process path, user, query, response, and the type of operation. On Windows and Linux endpoints where eBPF is enabled, DNS events are only recorded when you use the system DNS client. DNS events for applications that use their own DNS client are not recorded. For example, nslookup and Google Chrome.

HTTP HEADERS

[Windows only] Returns HTTP header information from connections on endpoints including the time connections occurred, the connection ID of the connection, the remote (destination) address, and the header name and value.

SYSTEM EVENTS

System level events capture when a user logs in or off.

System events provide common client extension infrastructure to other Tanium modules to notify them when a user logs in or off. This has the effect of the recorder pipeline being ‘on’ by default on Windows endpoints (connecting to ETW). System events are not stored in a database without a Threat Response recorder configuration.

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.

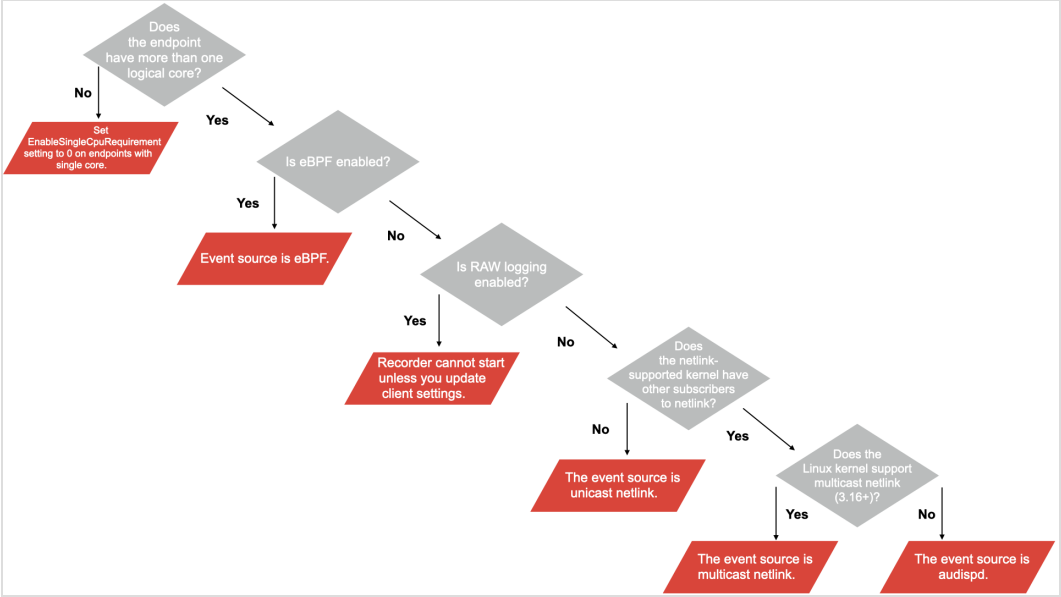
Some features of the Client Recorder Extension require specific versions of Windows.

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
Process hashes and command-line information	Tanium driver	Tanium driver	Tanium driver	Tanium driver	Tanium driver	Tanium driver
Driver loads	Available	Available	Available	Available	Available	Available

Sources of Client Recorder Extension data on Linux

The Client Recorder Extension for Linux uses netlink and the Linux audit subsystem to collect events. Auditd needs to be installed but is not required to be running for netlink unicast or multicast mode. Use the `Client Extensions - Status` sensor to check the event source for a Linux endpoint.



Extended Berkeley Packet Filter (eBPF)

eBPF is an open source kernel module framework that generates instrumentation in the kernel. It provides stable performance and has not been attributed to crashes. Fundamental to the architecture of this framework is the inability to loop and consequently it cannot cause endpoints to become unresponsive. This kernel module creates a new data source for Linux events.

eBPF provides an event source for process, file, and network events and serves as a replacement for auditd. Security events continue to come from auditd - however no rules need to be added to auditd. Security events that continue to come from auditd include:

- Anomalous Failure Messages
- Anomalous System or File Access
- Credential Acquired
- Credential Disposed
- Credential Refreshed
- Group Account Changes
- Group Added
- Group Deleted
- Group ID Changed
- Group Password Auth
- Labeled Security Protection Profile Events
- SE Linux User Error
- System Changes
- User Account Lock Changes
- User Account Modified
- User Added
- User Auth Token Modified
- User Authenticated
- User DAC Check
- User Deleted
- User Login
- User Logout
- User Management Data
- User Session Ended
- User Session Started
- User Space Hotplug Device
- User Space Messages

By default, eBPF is enabled on supported operating systems. To disable eBPF use the `CX.recorder.BPFEnableBPF` configuration setting. On endpoints with RHEL and CentOS versions 7.8 to 8.2 and OEL versions 8.3+, a BCC library is added to the endpoint. The Recorder eBPF program is compiled on the endpoint when eBPF is enabled. This program is recompiled every time the endpoint is restarted.

Kernel Driver (kaudit)

The kaudit process is a part of the Linux kernel responsible for the kernel audit events and forwards 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/`.

Client Recorder Extension by default does not load the Tanium rules if raw logging is enabled. If you are upgrading from Client Recorder Extension 2.1 and have Raw Logging enabled, the 2.2 upgrade removes the Tanium rules.

Netlink

Netlink is a mechanism provided by the Linux kernel to communicate between userspace and kernel space. Netlink enables the Linux Recorder to record event data directly to the Linux Kernel bypassing the auditd and audispd processes. The netlink socket can operate in unicast mode on all kernels, or in multicast mode on kernel versions 3.16 and later. When the Linux Recorder starts, it attempts to connect to the netlink socket in unicast mode, then multicast mode. If it is unable to connect with netlink it will fall back to the audispd plugin and TaniumAuditPipe. Subscribing to netlink in multicast or unicast mode performs more efficiently on the endpoint than the audispd plugin and TaniumAuditPipe. Auditd is installed and not running. Recorder uses the auditd libraries, but does not require auditd to be running.

If netlink connects successfully, the audispd plugin and TaniumAuditPipe are removed. On new kernels Recorder installations should automatically use netlink on their first boot.

There are two client configuration options for configuring netlink. If either of these options are toggled, the recorder functions in an ‘only connect via netlink’ configuration.

CX.recorder.AuditdEnableAudispdFallback

Allow getting events from audispd/TaniumAuditPipe. Run the `Recorder - Set Recorder Extension Setting [Linux]` package to enable this setting.

Default value: 1

CX.recorder.AuditdStopAuditdService

Stops the auditd service. For example, stops the auditd service so recorder can use the unicast or multicast netlink socket. Run the `Recorder - Set Recorder Extension Setting [Linux]` package to enable this setting.

Default value: 0

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/`.

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

Getting started

Step 1: Install the Client Recorder Extension

Install the Client Recorder Extension.

For more information, see Installing the Client Recorder Extension on page 20.

Step 2: Configure endpoint settings.

Configure endpoint settings.

For more information, see Configuring recorded events on page 25.

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: • 7.2.314.3476 (Linux, MacOS, Windows) • 7.2.314.3632 (Linux, MacOS, Windows) • 7.4.1.1955 and later For more information about specific Tanium Client versions, see Client Management User Guide: Client host system requirements.
Tanium products	One or more of the following Tanium products: • 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 amount of free disk space that is required depends on the configuration of the Client Recorder Extension. 3GB is recommended.

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.

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 to `/etc/audit/rules.d/tanium.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.
- Be aware that when using the failure "-f 2" mode, the Linux kernel panics in the event that auditd message is lost. The recorder does not add audit rules if this configuration is detected.

On SUSE 15 endpoints, make sure that `/etc/audit/rules.d/audit.rules` does not contain a rule added by default to suppress syscall auditing that displays as `-a task,never`. If this rule exists, remove it to ensure that events are recorded.

If SELinux is available and enforcing, the Client Recorder Extension attempts to install a policy when the Client Recorder Extension is installed or upgraded. If this policy is not installed - or not applied correctly - the following health check is returned:

```
SELinux is in enforcing mode but TaniumAuditPipe does not have the recorder SELinux policy applied.
```

If this health check is encountered, ensure that the `semodule`, `restorecon`, and `semanage` binaries are installed. Typically these binaries are installed in the same package; `policycoreutils` or `policycoreutils-python`. You can verify if these packages are installed by running a command such as `yum provides <path>/semodule`,

CO-RE is an abbreviation for Compile Once, Run Everywhere. When a kernel supports CO-RE, it can use the LibBPF library to configure BPF reporting. eBPF is supported on the following operating systems:

Operating system	Operating system version	eBPF supported	Notes
Oracle Enterprise Linux	8.3+ Standard Linux Kernel	Yes - LibBPF (CO-RE)	
	8.3+ UEK Kernel	Yes - BCC[1]	Kernel-uek-devel
	7.8-8.2 Standard Linux Kernel	Yes - BCC[2]	Kernel-devel, Kernel-headers
	7.8-8.2 UEK Kernel	Yes - BCC[1]	Kernel-uek-devel
RHEL 8, CentOS 8	8.2+	Yes - LibBPF (CO-RE)	

Operating system	Operating system version	eBPF supported	Notes
RHEL 7, CentOS 7	7.8-8.1	Yes - BCC[2]	Kernel-devel, Kernel-headers
Ubuntu	18.04-20.04	Yes - BCC[3]	linux-headers
Amazon Linux 2	2.0+	Yes - BCC[2]	Kernel-devel, Kernel-headers
SUSE, OpenSUSE, SUSE Linux Enterprise Server	12.4+	Yes - BCC[2]	Kernel-devel, Kernel-headers

[1] = with Kernel-uek-devel installed

[2] = with Kernel-devel and Kernel-headers installed

[3] = with linux-headers installed

The kernel-headers package and kernel-devel package can be installed with YUM. The version of the packages must match the version of the running kernel:

```
yum install kernel-devel-$(uname -r)
```

```
yum install kernel-headers-$(uname -r)
```

If running the UEK kernel on OEL, install the kernel-uek-devel package:

```
yum install kernel-uek-devel-$(uname -r)
```

The linux-headers package (Ubuntu 18.04 and later) can be installed with APT. The version of the package must match the version of the running kernel:

```
sudo apt install -y linux-headers-$(uname -r)
```

The debugfs file system is required. By default this is mounted under `sys/kernel/debug`. Make sure that `sys/kernel/debug` is not unmounted. If you are building a custom kernel, make sure that the DEBUG_FS option is enabled.

To determine if endpoints support using eBPF, use the Recorder - Is BPF Supported Details sensor. See Resolve details returned by the Recorder - Is BPF Supported Details sensor on page 40 for more information.

Disable raw logging before running the Client Recorder Extension on any Linux endpoint.

The Client Recorder Extension does not start on endpoints with a single logical core without updating the **CX.recorder.EnableSingleCpuRequirement** configuration setting to 0. To update CX.recorder.EnableSingleCpuRequirement to 0, edit the **Recorder - Set Recorder Extension Setting [*OS*]** package to add a parameter with the configuration key `EnableSingleCpuRequirement` and a value of `0`, and deploy the package to appropriate endpoints. Alternatively, you can run the following command from the Tanium Client directory on endpoints to update this configuration setting:

- (Windows) `TaniumClient.exe config set CX.recorder.EnableSingleCpuRequirement 0`
- (Linux and macOS) `./TaniumClient config set CX.recorder.EnableSingleCpuRequirement 0`

The Tanium Event Recorder Driver records 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 11
- Windows Server 2016
- Windows Server 2019
- Windows Server 2022

If the Tanium Event Recorder Driver is updated, endpoints require a reboot to ensure that all events are returned, to see the process tree in an alert, and to ensure that signals are working as intended. If you are deploying the 3.x version of Tanium Event Recorder Driver to endpoints for the first time, a reboot of endpoints is not required for the driver to capture events, but a reboot is required to view complete process tree data. If you are upgrading from Tanium Event Recorder Driver 2.x versions, endpoints require a reboot.

Support for macOS is the same as Tanium Client support.

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

Notes:

- Windows 7 and Windows Server 2008 R2 operating systems must have the following Microsoft KBs installed:
  - KB3033929 - "Availability of SHA-2 code signing support for Windows 7 and Windows Server 2008 R2." For details regarding KB3033929, see https://support.microsoft.com/en-us/help/3033929/microsoft-security-advisory-availability-of-sha-2-code-signing-support.

NOTE

- KB4490628 - "Servicing stack update for Windows 7 SP1 and Windows Server 2008 R2 SP1." For details regarding KB4490628, see https://support.microsoft.com/en-us/topic/servicing-stack-update-for-windows-7-sp1-and-windows-server-2008-r2-sp1-march-12-2019-b4dc0cff-d4f2-a408-0cb1-cb8e918feeba.
- KB4474419 - "SHA-2 code signing support update for Windows Server 2008 R2, Windows 7, and Windows Server 2008." For details regarding 4474419, see https://support.microsoft.com/en-us/topic/sha-2-code-signing-support-update-for-windows-server-2008-r2-windows-7-and-windows-server-2008-september-23-2019-84a8aad5-d8d9-2d5c-6d78-34f9aa5f8339.

- The recorder forces a vacuum if the database size becomes too large to ensure that a continual vacuuming does not exist. A check to only vacuum once per day and at least one hour after system startup to make sure vacuum operations do not interfere with system boot.

Host and network security requirements

Security exclusions

If security software is in use in the environment to monitor and block unknown host system processes, Tanium recommends that a security administrator create exclusions to allow the Tanium processes to run without interference. The configuration of these exclusions varies depending on AV software. For a list of all security exclusions to define across Tanium, see Tanium Core Platform Deployment Reference Guide: Host system security exclusions.

Target Device	Process
Module Server	*<Tanium Module Server>*\services\<ProductName>\node.exe
Windows endpoints	*<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
Linux endpoints	*<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
	<Tanium Client>/extensions/recorder/libTanuimBPF.so
	<Tanium Client>/extensions/recorder/TaniumAuditPipe
	/etc/audit/rules.d/tanium.rules
macOS endpoints	*<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

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)
`/Library/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)
`/Library/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)
`/Library/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)
`/Library/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)
`/Library/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)
`/Library/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)
`/Library/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)
`/Library/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`.

`/opt/Tanium/TaniumClient/extensions/recorder/selinux/tanium_client.pp` (Linux)
`/opt/Tanium/TaniumClient/extensions/recorder/selinux/tanium_recorder.pp` (Linux)

SELinux policies applied for Tanium Recorder and the Tanium Client when SELinux is enabled.

`/opt/Tanium/TaniumClient/extensions/recorder/libTaniumBPF.so` (Linux)

The library for Tanium Recorder eBPF interactions.

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

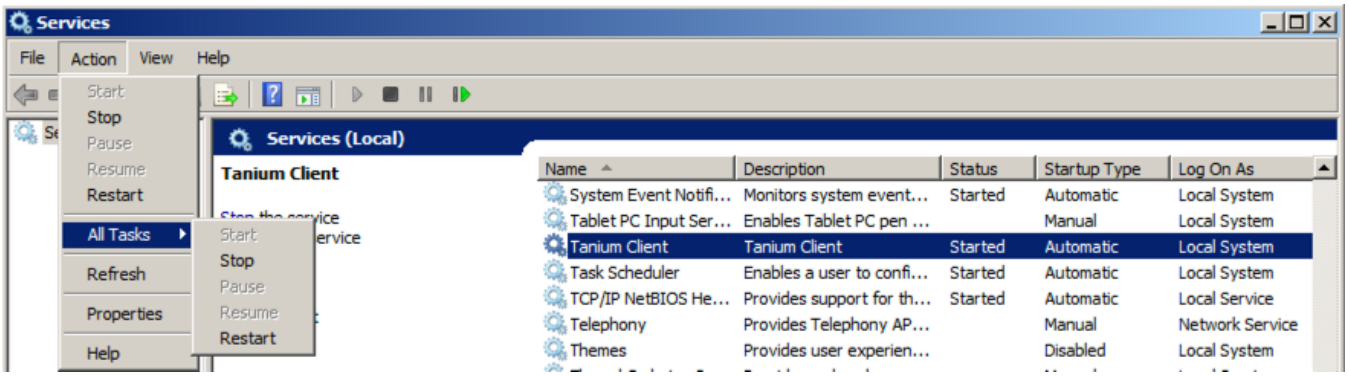
You might need to manually start or stop the Client Recorder Extension. For example, when troubleshooting you must resolve the underlying issue first and then manually restart the Client Recorder Extension. Or, if you find that the Client Recorder Extension is using more system resources than expected, 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 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 and later) `systemctl stop taniumclient` (Version 16 and later) `service taniumclient start` (Versions 14, 10) `service taniumclient stop` (Version 14, 10)

What to do next

See Getting started on page 13 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. Every subscription registered to the Client Recorder extension is saved in the recorder directory as a json file. Beginning with Client Recorder extension 2.2, if the subscription contains any signals Intel, this JSON file is encrypted.

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.

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

To understand how to apply Client Recorder Extension configuration settings, refer to any “Recorder” packages installed alongside the Client Recorder Extension. These packages demonstrate how to provide commands. For example ‘Recorder - Clear Subscriptions [Windows]’ demonstrates the use of the recorder command ‘recorder.uninstall’.

CX.recorder.AudispdMaxRestarts

Minimum audispd max restarts.

Default value: 10

CX.recorder.AudispdMaxRestartsWrite

Allow the recorder to use the value set in AudispdMaxRestarts to update audispd.conf

Default value: 0

CX.recorder.AudispdOverflowAction

audispd overflow action.

CX.recorder.AudispdOverflowActionWrite

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

Default value: 0

CX.recorder.AudispdPriorityBoost

Minimum audispd priority_boost.

Default value: 4

CX.recorder.AudispdPriorityBoostWrite

Allow the recorder to use the value set in AudispdPriorityBoost to update audispd.conf

Default value: 0

CX.recorder.AudispdQueueDepth

Minimum audispd q_depth.

Default value: 32768

CX.recorder.AudispdQueueDepthWrite

Allow the recorder to use the value set in AudispdQueueDepth to update audispd.conf

Default value: 1

CX.recorder.AuditDMaxWatchedPaths

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

Default value: 50

CX.recorder.AuditdDisableKernelPanicProtection

Allow recorder to run when auditd is in panic failure mode.

Default value: 0

CX.recorder.AuditdEnableAudispdFallback

Allow getting events from audispd/TaniumAuditPipe. Run the `Recorder - Set Recorder Extension Setting [Linux]` package to enable this setting.

Default value: 1

CX.recorder.AuditdEnableAuditd

The Client Recorder Extension connects to auditd by default. Disabling this setting results in the inability to record security events.

Default value: 1

CX.recorder.AuditdEnableAuditdConfMigration

Allow automatic migration of older auditd configs that prevent auditd from working properly with the recorder

Default value: 1

CX.recorder.AuditdEnableForkTracking

Enable/Disable tracking of forks

Default value: 1

CX.recorder.AuditdEnableIncomingNetwork

Enable Auditd incoming Network Monitoring. 0 is disabled

Default value: 1

CX.recorder.AuditdEnableNetlink

Enable Auditd Netlink

Default value: 1

CX.recorder.AuditdEnableOutgoingNetwork

Enable Auditd outgoing Network Monitoring. 0 is disabled

Default value: 1

CX.recorder.AuditdEnableRawLoggingRuleLoading

Enable/Disable loading of tanium rules if raw logging is detected on

Default value: 0

CX.recorder.AuditdEnableResolveLocalAddress

Enable/disable resolving local network address. 0 is disabled.

Default value: 1

CX.recorder.AuditdRawLogging

Allow recorder to update auditd.conf raw logging configuration.

AuditdRawLogging (1) = Recorder will turn ON raw logging in auditd.conf. Recorder will restart auditd if a setting is changed in auditd.conf.

AuditdRawLogging (0) = Recorder will turn OFF raw logging in auditd.conf. Recorder will restart auditd if a setting is changed in auditd.conf.

Default value: 0

CX.recorder.AuditdRawLoggingWrite

Allow Recorder to set the value that is configured in AuditdRawLogging. This must be set to 1 in order for recorder to use the value defined in AuditdRawLogging. If AuditdRawLoggingWrite is set to 0, any configuration set in AuditdRawLogging will not be used.

If another tool is being used to manage auditd.conf, these AuditdRawLoggingWrite should either be not defined or set to 0 so that Tanium Recorder does not modify auditd.conf

Default value: 0

CX.recorder.AuditdRulesBufferSize

Buffer size set in audit rules

Default value: 32768

CX.recorder.AuditdRulesDisableWatchSpecificPaths

Watch specific paths driven by subscription in auditd.

Default value: 0

CX.recorder.AuditdStopAuditdService

Stops the auditd service. For example, stops the auditd service so recorder can use the unicast or multicast netlink socket. Run the `Recorder - Set Recorder Extension Setting [Linux]` package to enable this setting.

Default value: 0

CX.recorder.BPFEnableBPF

Enables eBPF as an event source on Linux endpoints. 0 is disabled

Default value: 1

CX.recorder.ConfigRefreshConfigIntervalMins

How often to forceably refresh operating system config state.

Default value: 60

CX.recorder.ConfigRefreshIntervalSecs

How often to refresh configuration updates in seconds.

Default value: 300

CX.recorder.CpuThrottleCalculateTotalSystem

Calculate CPU utilization as a function of total system capacity not a single CPU

Default value: 1

CX.recorder.CpuThrottleMaximumSampleMilliseconds

Maximum time (ms) between samples for the CPU throttle check

Default value: 5000

CX.recorder.CpuThrottleMinimumSampleMilliseconds

Minimum time (ms) between samples for the CPU throttle check

Default value: 250

CX.recorder.CpuThrottleTargetPercent

Target maximum CPU (% total system capacity) for the extensions process

Default value: 5

CX.recorder.DatabaseChunkSize

Size of chunks in MB allocated from the filesystem when the database size increases.

Default value: 100

CX.recorder.DatabaseCleanContinueIntervalMs

Database Cleaning millisecond interval between cleaning each table.

Default value: 100

CX.recorder.DatabaseCleanIntervalSecs

Interval in seconds for the amount time to wait between each database cleaning operation.

Default value: 300

CX.recorder.DatabaseMaxCleanPercent

Maximum percent of the database to clean per cleaning. (50) for 50%

Default value: 50

CX.recorder.DatabaseMaxSizeMB

Max size of the database before cleaning in MB.

Default value: 1024

CX.recorder.DatabaseMinCleanPercent

Minimum percent of the database to clean per cleaning. This will clean at least this amount each time the database is over the target size.

Default value: 5

CX.recorder.DatabaseReset

Delete database on startup this setting will reset automatically when database has been resetted.

Default value: 0

CX.recorder.DatabaseStatsUpdateSecs

Number of seconds between updating the database statistics used in metrics and cleaning.

Default value: 60

CX.recorder.DeleteLegacyDatabaseAfterDays

Time to wait since the recorder database is created before deleting legacy database

Default value: 14

CX.recorder.DeleteLegacyDatabaseFreeMB

Required free space to be available to keep legacy monitor.db around.

Default value: 1024

CX.recorder.DigestBufferSizeKB

Buffer size for Index file read and digest calculation in KB

Default value: 64

CX.recorder.DisableResourceMonitor

Disable resource monitor

Default value: 0

CX.recorder.EmitFileWriteIntervalSecs

The interval between emitting file write events for the same file handle.

Default value: 30

CX.recorder.EnableEtw

Enable Windows ETW events.

Default value: 1

CX.recorder.EnableEtwDns

Enable Windows ETW for DNS Events

Default value: 1

CX.recorder.EnableEtwFile

Enable Windows ETW for File Events

Default value: 1

CX.recorder.EnableEtwNetwork

Enable Windows ETW for Network Events

Default value: 1

CX.recorder.EnableEtwProcess

Enable Windows ETW for Process Events

Default value: 1

CX.recorder.EnableEtwRegistry

Enable Windows ETW for Registry Events

Default value: 1

CX.recorder.EnableEtwSysmon

Enable Windows ETW for Sysmon Events

Default value: 0

CX.recorder.EnableEtwTaniumDriver

Enable Windows ETW for Tanium Driver Events

Default value: 1

CX.recorder.EnableEvt

Enable Windows Eventlog events.

Default value: 1

CX.recorder.EnableKEtw

Enable Windows Kernel ETW events.

Default value: 1

CX.recorder.EnableLibraryHashing

Enabling hashing of library load events.

Default value: 1

CX.recorder.EnableMacBSM

Enable BSM on macOS.

Default value: 1

CX.recorder.EnableMacEXESigCheck

Enable checking executables for signatures on macOS.

Default value: 1

CX.recorder.EnableMacFSEvents

Enable FSEvents on macOS.

Default value: 1

CX.recorder.EnableMacKext

Enable kext load events on macOS.

Default value: 1

CX.recorder.EnableMacPKTAP

Enable PKTAP on macOS.

Default value: 1

CX.recorder.EnableMisconfiguredAuditPolicyHealthCheck

Threat Response profiles for Windows endpoints enable capturing specific sets of event IDs. Sometimes client audit policies can not be configured properly to ensure all relevant events are generated. The EnableMisconfiguredAuditPolicyHealthCheck setting provides a way to detect when this happens. By default, this setting is disabled. Clients generate a health check when this option is enabled. When viewing the logs, you may see content where a single event id might actually apply to multiple audit policy categories. It is possible in some situations to see what appears to be unrelated audit policy check failures.

Default value: 0

CX.recorder.EnableSignalSubscriptionProtection

Enable data protection for signal intel subscriptions.

Default value: 1

CX.recorder.EnableSubscriptionProtection

Enable data protection for recorder subscriptions.

Default value: 0

CX.recorder.EnableWinDLLSigCheck

Enable checking all loaded DLLs for signatures on Windows.

Default value: 1

CX.recorder.EnableWinEXESigCheck

Enable checking executables for signatures on Windows.

Default value: 1

CX.recorder.EnableWinSYSSigCheck

Enable checking loaded drivers for signatures on Windows.

Default value: 1

CX.recorder.EtwMatchingToleranceMs

The milliseconds of tolerance allowed in ETW when matching events to a process.

Default value: 100

CX.recorder.EventAssemblerTimeoutMs

Number of MS assembler will wait to complete a process context before forwarding.

Default value: 10000

CX.recorder.FSEventsIgnoreOwnEvents

Ignore Recorder-generated file events on macOS

Default value: 1

CX.recorder.FSEventsQueueSize

FSEvents queue size on macOS.

Default value: 512

CX.recorder.FSEventsUseDefaultExclusions

Ignore default nuisance paths on macOS

Default value: 1

CX.recorder.FileInfoProviderDomain

Domain of FileInfoProvider used to hash executables

CX.recorder.FileInfoProviderName

Name of FileInfoProvider used to hash executables

CX.recorder.HashCacheSize

Size of hash cache in Processor hash provider

Default value: 50

CX.recorder.HealthLogWriteIntervalMins

Interval to write health issues to logfile for polled items.

Default value: 30

CX.recorder.JournaldRawLogging

Enable/Disable systemd journald auditd socket raw logging

Default value: 0

CX.recorder.JournaldRawLoggingWrite

Allow the recorder to use the value set in JournaldRawLogging to call systemctl

Default value: 0

CX.recorder.MailboxQueryTimeoutMs

The timeout for recorder mailbox query commands.

Default value: 60000

CX.recorder.MaxHashSizeMB

Max file size in MB to hash

Default value: 32

CX.recorder.MaxMailboxItemsPerCheck

How many inbox messages to consume per work item.

Default value: 32

CX.recorder.PathReassemblyTimeoutSecs

The number of seconds the event converter will wait to assemble a path before deleting.

Default value: 300

CX.recorder.PersistenceQueueDrainIntervalMs

How often to check the queue for processed events

Default value: 150

CX.recorder.PersistenceWriteProcessIntervalSecs

How often to periodicially write the proc.bin file to disk

Default value: 600

CX.recorder.ProcessorCacheWriteIntervalSecs

Interval to write process hash and signature hash to disk in seconds

Default value: 300

CX.recorder.ProcessorCleanupIntervalSecs

How often to perform cleanups in the processor

Default value: 15

CX.recorder.ProcessorPruneAdjustmentSecs

Additional tolerance required before the processing stage prunes caches

Default value: 10

CX.recorder.ProcessorQueueDrainIntervalMs

How often to check the queue for parsed events

Default value: 10

CX.recorder.ReconfigureIntervalSeconds

How often to check for configuration changes

Default value: 60

CX.recorder.ResourceMonitorCPUPercent

Sustained CPU percentage required to suspend subscriptions

Default value: 50

CX.recorder.ResourceMonitorRetryAttempts

Number of times resource monitor can cause a temporary suspension

Default value: 0

CX.recorder.ResourceMonitorRetryBackoffTimeSecs

Time in seconds subscriptions will be temporarily suspended before reenabling

Default value: 600

CX.recorder.ResourceMonitorTrackedWindowMins

Window of time that a temporary suspension counts towards a sticky suspension

Default value: 1440

CX.recorder.SnapshotExtraSizeRequiredMB

Required extra free space to be available to perform a snapshot.

Default value: 1024

CX.recorder.SnapshotWriteIntervalMs

The interval that snapshots will write pages to the destination database.

Default value: 100

CX.recorder.SnapshotWriteSizeMB

The size of each write, in megabytes, that the snapshot will write each SnapshotWriteIntervalMs interval.

Default value: 24

CX.recorder.Stage1WorkQueueInterval

How often to pull items from the stage one worker queues in milliseconds.

Default value: 10

CX.recorder.StatusUpdateInterval

How often to update status information.

Default value: 60

CX.recorder.SyslogPluginRawLogging

Enable/Disable raw logging via the syslog auditd plugin

Default value: 0

CX.recorder.SyslogPluginRawLoggingWrite

Allow the recorder to use the value set in SyslogPluginRawLogging to update plugins.d/syslog.conf

Default value: 0

CX.recorder.VacuumIntervalDays

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

The recorder forces a vacuum if the database size becomes too large to ensure that a continual vacuuming does not exist. A check to only vacuum once per day and at least 1 hr after system startup to make sure vacuum operations do not interfere with system boot.

Default value: 7

CX.recorder.EnableSingleCpuRequirement

The Client Recorder Extension does not start on endpoints with a single CPU without updating the CX.recorder.EnableSingleCpuRequirement configuration setting to 0.

Default value: 1

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 Threat Response 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.

   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.

   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.

Resolve details returned by the Recorder - Is BPF Supported Details sensor

To determine if endpoints support eBPF as an event source, you can use the Recorder - Is BPF Supported Details sensor. The following table provides an explanation of the possible results the sensor returns and possible resolutions when the sensor returns false for any key.

Key	Description	Possible cause and potential remediation for false
BPF Headers found	If true, this indicates that headers were detected on the endpoint that are required to run BPF.	If not found, the kernel is either too old or compiled without BPF support explicitly.
Kernel Headers match running kernel version	If true, this indicates that the requirement for the version of the kernel-headers package matching the version of the kernel running has been met.	The newest kernel-headers have been installed on the endpoint, but the endpoint is running an old kernel. To remediate this cause, a reboot is recommended. This could also indicate that there are no headers installed.

Key	Description	Possible cause and potential remediation for false
Kernel-devel package patch running kernel version	If true, this indicates that the requirement for the version of the kernel-devel package matches the version of the kernel running.	The newest kernel-devel has been installed on the endpoint, but the endpoint is running an old kernel. To remediate this cause, a reboot is recommended. This could also indicate that there are no headers installed.
Possible BTF Support	This kernel supports running in libbpf mode, and is most likely a new endpoint (RHEL 8.2+, CentOS 8.2+, Ubuntu 20.10+).Recorder runs in libbpf mode on this endpoint.	If false, kernel is probably too old to support LibBPF.
Supported Operating System	Recorder supports running BPF (either mode) on this OS / distribution. RHEL is supported, Amazon Linux (although RHEL based) is currently not.	The operating system is not supported.
Supported Platform	The platform is supported. Linux, returns true. Anything else should be false.	The platform is not supported.
Kernel Without Confidentiality Mode Enabled	BPF cannot run on OEL endpoints if the kernel is set to confidentiality mode.	Confidentiality mode in OEL prevents BPF programs from running.