Streaming Telemetry

- Information About Streaming Telemetry , on page 1
- Gather Points, on page 1
- Subscription, on page 2
- Transport , on page 3
- Scale Considerations , on page 3
- Session, on page 3
- Configuring Telemetry on a Cisco Catalyst 9800 Series Wireless Controller, on page 4
- On-Change Telemetry Support , on page 11
- Supported XPaths for On-Change Subscription, on page 11
- Troubleshooting Telemetry Support, on page 15
- Cisco Catalyst Center Client Event and SSID Telemetry Filter, on page 17

Information About Streaming Telemetry

Streaming telemetry is a new paradigm in monitoring the health of a network. It provides a mechanism to efficiently stream configuration and operational data of interest from the Cisco Catalyst 9800 Series Wireless Controller. This streamed data is transmitted in a structured format to remote management stations for monitoring and troubleshooting purposes.

This topic explains how to enable the telemetry support the Wi-Fi and system health-related data. Not that telemetry support can be enhanced up to a scale of 1000 access points (APs) and 15000 clients. A single collector setup can be used to subscribe to the requested XPaths. A telemetry feed can be used to subscribe to data elements to monitor APs and clients effectively. Data is provided through the native Cisco wireless models.

Gather Points

Gather points are the top-level XPaths and act as the smallest unit of data exported by a target. Any subscription to an XPath raises to the level of the Gather point, and the target sends updates comprising of all the leaves defined under this Gather point. For example, when you subscribe to an XPath */access-point-operdata/radio-oper-data/vap-oper-config/ssid*, which is part of the Gather point */access-point-operdata/radio-oper-data/vap-oper-config*, the reply will comprise of all the attributes that are a part of the Gather point, in this case, AP-VAP-ID, SSID, and WLAN ID.

The following lists the supported Gather points for an XPaths.

Table 1: Supported Gather Points and Subscription Intervals

Supported Gather Point	Subscription Interval
wireless-access-point-oper:access-point-oper-data/ethernet-mac-wtp-mac-map	>=15 mins
/wireless-access-point-oper:access-point-oper-data/capwap-data	>=15 mins
/wireless-access-point-oper:access-point-oper-data/cdp-cache-data/	>=15 mins
/wireless-access-point-oper:access-point-oper-data/radio-oper-stats	>=60 secs
/wireless-access-point-oper:access-point-oper-data/radio-oper-data	>=180 secs
/wireless-access-point-oper:access-point-oper-data/oper-data	>=180 secs
/wireless-rrm-oper:rrm-oper-data/rrm-measurement	>=180 secs
/wireless-client-oper:client-oper-data/dot11-oper-data	>=180 secs
/wireless-client-oper:client-oper-data/common-oper-data	>=15 mins
/wireless-client-oper:client-oper-data/policy-data	>=60 secs
/wireless-client-oper:client-oper-data/sisf-db-mac/ipv4-binding/ip-key/ip-addr	>=15 mins
/wireless-client-oper:client-oper-data/traffic-stats	>=180 secs
/lldp-ios-xe-oper:lldp-entries/lldp-state-details	>=60 secs
/device-hardware-xe-oper:device-hardware-data/device-hardware	>=15 mins
/wireless-mobility-oper:mobility-oper-data/mobility-node-data/ulink-status	>=60 secs
/process-cpu-ios-xe-oper:cpu-usage/cpu-utilization/one-minute	>=60 secs
/platform-sw-ios-xe-oper:cisco-platform-software/control-processes	>=60 secs
/environment-ios-xe-oper:environment-sensors/environment-sensor	>=60 secs
/lldp-ios-xe-oper:lldp-entries/lldp-intf-details	>=60 secs
/interfaces-ios-xe-oper:interfaces/interface	>=60 secs
/platform-ios-xe-oper:components/component	>=60 secs
/mdt-oper-v2:mdt-oper-v2-data	>=60 secs
/wireless-access-point-oper:access-point-oper-data/radio-oper-data/radio-band-info	>=180 secs

Subscription

A subscription binds one or more Gather points and destinations. A Multicast Default (MDT) streams data for each Gather point at the configured frequency (cadence-based streaming).

Transport

The protocol that is used for the connection between a publisher and a receiver is known as the transport protocol, and this decides how data are transmitted. This protocol is independent of the management protocol for configured subscriptions. The supported transport protocols are gNMI and gRPC. The gNMI transport protocol supports JSON encoding of data, while gRPC supports Key-value Google Protocol Buffers (kvGPB) encoding.

Scale Considerations

The following table provides the scale numbers that are applicable to the native model for an XPath set.

Table 2: Scaling Considerations to the Native Model

Attribute	Scale
AP	4000
Client	15000
SSID Per AP	6
BSSID per AP	12
Neighbors per AP	60 (30x2)
Number of Physical Neighbor APs	49
Number of Neighbor Records	60000 records

Session

You can choose to initiate the subscription by establishing a telemetry session between the controller and the receiver. A telemetry session can be initiated using:

- gNMI Dial-In Mode
- gRPC Dial-Out Mode

gNMI Dial-In-Mode

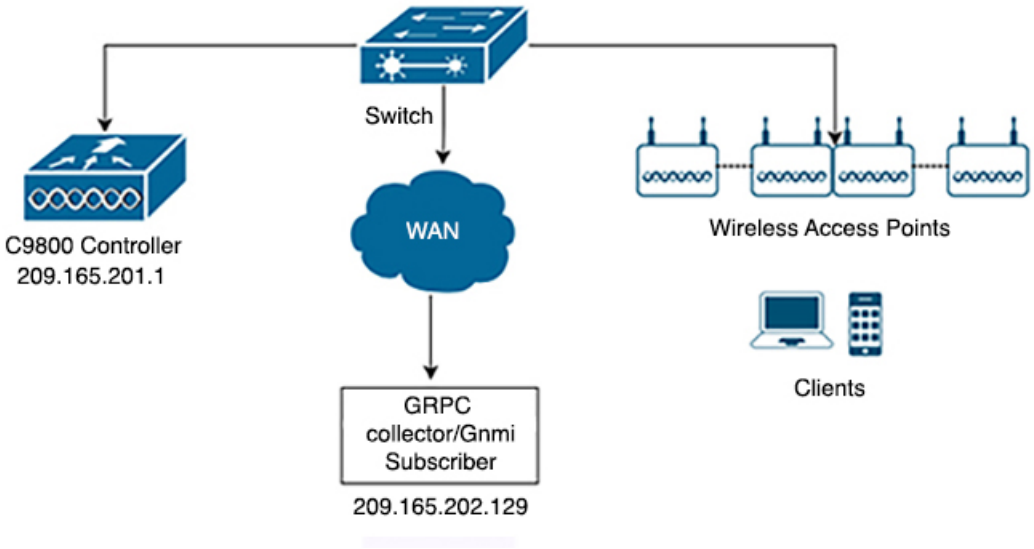
In a dial-in mode, a Model-Driven Telemetry (MDT) receiver dials in to the controller, and subscribes dynamically to one or more Gather points or subscriptions. The controller acts as the server, and the receiver as the client. The controller streams telemetry data through the same session. The dial-in mode of subscriptions is dynamic, which gets terminated when the receiver cancels the subscription or when the session is terminated.

gRPC- Dial-Out-Mode

In a dial-out mode, the controller dials out to the receiver. Here the controller acts as a client and receiver acts as a server. In this mode, Gather points and destinations are configured and bound together into one or more subscriptions. The controller continually attempts to establish a session with each destination in the subscription, and streams data to the receiver. The dial-out mode of subscriptions is persistent.

Figure 1: Telemetry Session

The following figure explains the telemetry session:



Configuring Telemetry on a Cisco Catalyst 9800 Series Wireless Controller

To configure telemetry on a Cisco Catalyst 9800 Series Wireless Controller, perform the following:

1. Enable gNXI in an Insecure Mode
2. Enable gNXI in a Secure Mode
3. Verify the Status of the Subscription
4. Manage Configured Subscriptions

Enabling gNXI in Insecure Mode (CLI)

Procedure

	Command or Action	Purpose
Step 1	**enable**	Enables privileged EXEC mode

	Command or Action	Purpose
	Example: `Device# enable`	Enter your password, if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**gnxi** **Example:** `Device(config)# gnxi`	Starts the gNXI process.
Step 4	**gnxi server** **Example:** `Device(config)# gnxi server`	Enables the gNXI server in insecure mode.
Step 5	**gnxi port** *port-number* **Example:** `Device(config)# gnxi 50000`	Sets the gNXI port. The default insecure gNXI port is 9339.
Step 6	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 7	**show gnxi state** **Example:** `Device# show gnxi state`	Displays the status of gNXI server.

Example

The following is a sample output of the **show gnxi state** command:

```
Device# show gnxi state
State Status
--------------------------------
Enabled Up
```

Enabling gNXI in Secure Mode (CLI)

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device# enable`	Enables privileged EXEC mode Enter your password, if prompted.

	Command or Action	Purpose
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**gnxi** **Example:** `Device(config)# gnxi`	Starts the gNXI process.
Step 4	**gnxi secure-server** **Example:** `Device(config)# gnxi secure-server`	Enables the gNXI server in secure mode.
Step 5	**gnxi secure-trustpoint** *trustpoint-name* **Example:** `Device(config)# gnxi secure-trustpoint`	Specifies the trustpoint and certificate set that gNXI uses for authentication.
Step 6	**gnxi secure-client-auth** **Example:** `Device(config)# gnxi secure-client-auth`	(Optional) The gNXI process authenticates the client certificate against the root certificate.
Step 7	**gnxi secure-port** **Example:** `Device(config)# gnxi secure-port`	(Optional) Sets the gNXI port. • The default insecure gNXI port is 9339.
Step 8	**end** **Example:** `Device(config)# end`	Returns to privileged EXEC mode.
Step 9	**show gnxi state** **Example:** `Device# show gnxi state`	Displays the gNXI servers status.

Example

The following is sample output from the **show gnxi state** command:

```
Device# show gnxi state
State Status
--------------------------------
Enabled Up
```

Verifying the Status of a Telemetry Subscription on a Cisco Catalyst 9800 Series Wireless Controller

To verify the status of a subscription, use the following command:

```
Device# show telemetry ietf subscription all
Device# show telemetry ietf subscription 101
Device# show telemetry ietf subscription 101 detail
Device# show telemetry ietf subscription 101 receiver
Device# show telemetry internal connection
Device# show telemetry internal subscription all stats
Device# show telemetry receiver all
Device# show telemetry receiver name <receivers-name>
Device# show telemetry connection all
```

Managing Configured Subscriptions on a Cisco Catalyst 9800 Series Wireless Controller

Use the **show platform software ndbman switch** {*switch-number* | **active**| **standby**} **models** command to display the list of YANG models that support on-change subscription.

Note Currently, you can only use the gRPC protocol for managing configured subscriptions.

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device# enable`	Enables privileged EXEC mode Enter your password, if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**telemetry ietf subscription** *id* **Example:** `Device(config)# telemetry ietf subscription 112`	Creates a telemetry subscription and enters telemetry-subscription mode.
Step 4	**encoding encode-kvgpb** **Example:** `Device(config-mdt-subs)# encoding encode-kvgpb`	Specifies the Key-value Google Protocol Buffers (kvGPB) encoding.

	Command or Action	Purpose
Step 5	**filter xpath** *path* **Example:** `Device(config-mdt-subs)# filter xpath /wireless-access-point-oper:access-point-oper-data/capwap-data`	Specifies the XPath filter for the subscription.
Step 6	**source-address**{ *A.B.C.D* | *X:X:X:X::X* } **Example:** `Device(config-mdt-subs)# source-address ip-address 209.165.200.225 | 2001:DB8::1`	Configures the source IP address on the telemetry subscription interface.
Step 7	**stream yang-push** *path* **Example:** `Device(config-mdt-subs)# stream yang-push`	Configures a stream for the subscription.
Step 8	**update-policy {on-change | periodic}** *period* **Example:** `Device(config-mdt-subs)# update-policy periodic 3000`	Configures a periodic update policy for the subscription.
Step 9	**receiver ip address** *ip-address receiver-port* **protocol** *protocol* **profile** *name* **Example:** `Device(config-mdt-subs)# receiver ip address 209.165.201.1 protocol grpc-tcp`	Configures a periodic update policy for the subscription.
Step 10	**end** **Example:** `Device(config-mdt-subs)# end`	Exits telemetry-subscription configuration mode and returns to privileged EXEC mode.

Zero Trust Telemetry

To configure zero trust telemetry on a Cisco Catalyst 9800 Series Wireless Controller, perform the following:

1. Define a protocol
2. Define a named receiver
3. Configure telemetry subscription

Define a Protocol

Before you begin

Define crypto trustpoints (CAforMDTserver and IDforWLCclient) and certificates before the telemetry configuration.

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 2	**telemetry protocol grpc profile** *profile-name* **Example:** `Device(config)# telemetry protocol grpc profile mtlsyang`	Configures the protocol gRPC profile and enters gRPC profile name.
Step 3	**ca-trustpoint** *ca-for-mdt-server* **Example:** `Device(config-mdt-protocol-grpc-profile)# ca-trustpoint CAforMDTserver`	Adds the server CA trustpoint.
Step 4	**id-trustpoint** *wlc-id-trustpoint* **Example:** `Device(config-mdt-protocol-grpc-profile)# id-trustpoint IDforWLCclient`	Adds the client ID trustpoint.

Define a Named Receiver

This procedure defines:

- FQDN DNS name
- Crypto protocol definition

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 2	**telemetry receiver protocol** *receiver-name* **Example:** `Device(config)# telemetry receiver protocol collector`	Configures the receiver name.

	Command or Action	Purpose
Step 3	**host name** *FQDN-receiver* **Example:** `Device(config-mdt-protocol-receiver)# host name collector-telemetry.cisco.com 57500`	Adds FQDN DNS name of receiver.
Step 4	**protocol grpc-tls profile** *profile-name* **Example:** `Device(config-mdt-protocol-receiver)# protocol grpc-tls profile mtlsyang`	Defines the gRPC TLS profile named mtlsyang.

Configure Telemetry Subscription

This procedure configures:

- Xpath
- Named receiver
- Protocol

Procedure

	Command or Action	Purpose
Step 1	**enable** **Example:** `Device# enable`	Enables privileged EXEC mode Enter your password, if prompted.
Step 2	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 3	**telemetry ietf subscription** *id* **Example:** `Device(config)# telemetry ietf subscription 113`	Creates a telemetry subscription and enters telemetry-subscription mode.
Step 4	**encoding encode-kvgpb** **Example:** `Device(config-mdt-subs)# encoding encode-kvgpb`	Specifies the Key-value Google Protocol Buffers (kvGPB) encoding.
Step 5	**filter xpath** *path* **Example:** `Device(config-mdt-subs)# filter xpath /wireless-ble-ltx-oper:ble-ltx-oper-data/ble-ltx-ap-streaming`	Specifies the XPath filter for the subscription.

	Command or Action	Purpose
Step 6	**source-address** { *A.B.C.D* | *X:X:X:X::X* } **Example:** `Device(config-mdt-subs)# source-address ip-address 209.165.200.225 | 2001:DB8::1`	Configures the source IP address on the telemetry subscription interface.
Step 7	**stream yang-push** **Example:** `Device(config-mdt-subs)# stream yang-push`	Configures a stream for the subscription.
Step 8	**update-policy {on-change | periodic}** *period* **Example:** `Device(config-mdt-subs)# update-policy periodic 6000`	Configures a periodic update policy for the subscription.
Step 9	**receiver-type protocol** **Example:** `Device(config-mdt-subs)# receiver-type protocol`	Configures type protocol for receiver.
Step 10	**receiver name** *receiver-name* **Example:** `Device(config-mdt-subs)# receiver name collector`	Specifies the receiver name.
Step 11	**end** **Example:** `Device(config-mdt-subs)# end`	Exits telemetry-subscription configuration mode and returns to privileged EXEC mode.

On-Change Telemetry Support

From Cisco IOS XE Cupertino 17.7.1 onwards, on-change telemetry support is provided to a subset of XPaths.

Supported XPaths for On-Change Subscription

The following table lists the supported XPaths for on-change subscription.

Table 3: Supported Gather Points and XPaths

Gather Points	XPaths
/access-point-oper-data/radio-operdata/	/access-point-oper-data/radio-operdata/ phy-ht-cfg/cfg-data/curr-freq

Gather Points	XPaths
	/access-point-oper-data/radio-operdata/phy-ht-cfg/cfg-data/chan-width
	/access-point-oper-data/radio-oper-data/current-band-id
/access-point-oper-data/capwap-data	/access-point-oper-data/capwap-data/name
	/access-point-oper-data/capwapdata/device-detail/wtp-version/sw-ver/version
	/access-point-oper-data/capwap-data/device-detail/wtp-version/sw-ver/release
	/access-point-oper-data/capwapdata/device-detail/wtp-version/sw-ver/maint
	/access-point-oper-data/capwapdata/device-detail/wtp-version/sw-ver/build
	/access-point-oper-data/capwap-data/ap-state/ap-operation-state
	/access-point-oper-data/capwapdata/device-detail/static-info/board-data/wtp-serial-num
/access-point-oper-data/oper-data	/access-point-oper-data/oper-data/ap-ip-data/ap-ip-addr
	/access-point-oper-dat/oper-data/ap-pow/power-type

The following table lists the XPaths that are introduced in Cisco-IOS-XE-wireless-ap-global-oper-transform.yang model that is displayed through telemetry feed.

Table 4: Supported Gather Points and XPaths (Cisco-IOS-XE-wireless-ap-global-oper-transform.yang)

Gather Points	XPaths
/ap-global-oper-data/ap-join-stats/wtp-mac	/ap-global-oper-data/ap-join-stats/ap-join-info/ap-ethernet-mac
	/ap-global-oper-data/ap-join-stats/ap-join-info/ap-name
	/ap-global-oper-data/ap-join-stats/ap-join-info/ap-ip-addr
	/ap-global-oper-data/ap-join-stats/ap-join-info/is-joined
	/ap-global-oper-data/ap-join-stats/ap-join-info/last-error-type
	/ap-global-oper-data/ap-join-stats/ap-disconnect-reason

The following table lists the XPaths that are introduced in Cisco-IOS-XE-aaa-oper.yang model to support aaa/radius/radsec and displayed through telemetry feed.

Table 5: Supported Gather Points and XPaths (Cisco-IOS-XE-aaa-oper.yang)

Gather Points	Xpaths
/aaa-data/aaa-radius-stats/	/aaa-data/aaa-radius-stats/radsec-pkt-cnt-idletime
	/aaa-data/aaa-radius-stats/radsec-send-hs-start-cnt
	/aaa-data/aaa-radius-stats/radsec-hs-success-cnt
	/aaa-data/aaa-radius-stats/radsec-total-tx-pkt-cnt
	/aaa-data/aaa-radius-stats/radsec-total-rx-pkt-cnt
	/aaa-data/aaa-radius-stats/radsec-total-conn-rst-cnt
	/aaa-data/aaa-radius-stats/radsec-conn-rst-cnt-idle /aaa-data/aaa-radius-stats/radsec-conn-rst-cnt-noresp
	/aaa-data/aaa-radius-stats/radsec-conn-rst-cnt-malpkt
	/aaa-data/aaa-radius-stats/radsec-conn-rst-cnt-err
	/aaa-data/aaa-radius-stats/radsec-conn-rst-cnt-peer
	/aaa-data/aaa-radius-stats/num-aaa-lib-inst
	/aaa-data/aaa-radius-stats/server-detail
/aaa-data/aaa-radius-global-stats	/aaa-data/aaa-radius-global-stats/access-rejects
	/aaa-data/aaa-radius-global-stats/access-accepts
	/aaa-data/aaa-radius-global-stats/authen-responses-seen
	/aaa-data/aaa-radius-global-stats/authen-with-response
	/aaa-data/aaa-radius-global-stats/authen-without-response
	/aaa-data/aaa-radius-global-stats/authen-avg-response-delay
	/aaa-data/aaa-radius-global-stats/authen-max-response-delay
	/aaa-data/aaa-radius-global-stats/authen-timeouts
	/aaa-data/aaa-radius-global-stats/authen-duplicate-id
	/aaa-data/aaa-radius-global-stats/authen-bad-authenticators
	/aaa-data/aaa-radius-global-stats/acct-responses-seen
	/aaa-data/aaa-radius-global-stats/acct-with-response

Gather Points	Xpaths
	/aaa-data/aaa-radius-global-stats/acct-without-response
	/aaa-data/aaa-radius-global-stats/acct-avg-response-delay
	/aaa-data/aaa-radius-global-stats/acct-max-response-delay
	/aaa-data/aaa-radius-global-stats/acct-timeouts
	/aaa-data/aaa-radius-global-stats/acct-duplicate-id
	/aaa-data/aaa-radius-global-stats/acct-bad-authenticators
	/aaa-data/aaa-radius-global-stats/stats-time

The following table lists the XPaths that are introduced in Cisco-IOS-XE-wireless-mesh-rpc.yang model to support the mesh-related EXEC commands:

Table 6: Supported EXEC CLIs and XPaths (Cisco-IOS-XE-wireless-mesh-rpc.yang)

EXEC CLI	XPath
ap name <*ap-name*> [no] mesh ethernet [0|1|2|3] mode trunk vlan allowed <*vlan-id*>	/set-rad-mesh-ethernet-trunk-allowed-vlan
ap name <*ap-name*> [no] mesh ethernet [0|1|2|3] mode trunk vlan native	/set-rad-mesh-ethernet-trunk-native-vlan
ap name <*ap-name*> mesh linktest <*dst AP MAC*> <*data rate*> <*packets/sec*> <*packet size*> <*duration*>	/exec-linktest-ap
ap name <*ap-name*> [no] mesh ethernet [0|1|2|3] mode access <*vlan-id*>	/set-rad-mesh-ethernet-access-vlan
ap name <*ap-name*> [no] mesh block-child	/set-rad-mesh-block-child
ap name <*ap-name*> [no] mesh vlan-trunking	/set-rad-mesh-trunking
ap name <*ap-name*> [no] mesh daisy-chaining strict-rap	/set-rad-mesh-daisy-chain-strict-rap
ap name <*ap-name*> [no] mesh daisy-chaining	/set-rad-mesh-daisy-chain-mode
ap name <*ap-name*> [no] mesh parent preferred	/set-rad-mesh-preferred-parent-ap
ap name <*ap-name*> mesh backhaul rate dot11ac mcs <*mcs-index*> *ss* <*1-4*>	/set-rad-mesh-bhaul-tx-rate
ap name <*ap-name*> mesh backhaul radio dot11 5ghz [slot <*slot-id*>]	/set-rad-mesh-bhaul-radio
ap name <*ap-name*> mesh security psk provisioning delete	/set-rad-mesh-security-psk-provisioning-delete

EXEC CLI	XPath
ap name <*ap-name*> mesh vlan-trunking native <*vlan-id*>	/set-rad-mesh-trunking-vlan

The following table lists the XPaths that are introduced in Cisco-IOS-XE-aaa-oper.yang model to support radius EXEC commands:

Table 7: Supported EXEC CLIs and XPaths (Cisco-IOS-XE-aaa-oper.yang)

EXEC CLIs	XPaths
show radius statistic	/aaa-data/aaa-radius-global-stats/

Troubleshooting Telemetry Support

This document outlines a set of commands for gathering data from Cisco Catalyst 9800 Series Wireless Controller, specifically focused on addressing gRPC telemetry-related issues in support of TAC cases.

Here are a few factors to consider when conducting troubleshooting steps:

- Provide a clear problem description.
- What has changed in the network?
- What was the previous working day/time?
- What is the impact of this problem?

Note Run all the **show** commands with **show clock** or **terminal exec prompt timestamp** once to log timestamps automatically.

General Guidelines

For every issue, run the following commands:

1. `Device# terminal length 0`
2. `Device# show clock`
3. `Device# show tech-support wireless`
4. `Device# request platform software trace archive last 1`

Perform Basic Checks

1. Verify that the requisite processes (particularly pubd) are running using the following commands:

   ```
   show platform software yang-management process
   ```

2. Capture and validate the telemetry-specific configuration using the following command:

```
show running-config | section telemetry
```

3. Check the validity of any subscriptions using the following command:

```
show telemetry ietf subscription all
```

4. Check the validity of any named receivers using the following command:

```
show telemetry receiver all
```

5. Verify the telemetry subscription states using the following command:

```
show telemetry internal subscription all stats
```

Check Connectivity Issues

1. Check the state of the subscription receiver using the following commands:

```
show telemetry ietf subscription <id> receiver
```

2. Check the state of telemetry connections using the following command:

```
show telemetry connection all
```

3. Check which subscriptions use a particular connection using the following command:

```
show telemetry connection <index> subscription
```

Capture Debug Logs

1. Enable the following debug options:

```
set platform software trace mdt-pubd chassis active r0 mdt-ctrl debug
set platform software trace mdt-pubd chassis active r0 pubd debug
set platform software trace mdt-pubd chassis active r0 green-be debug
set platform software trace mdt-pubd chassis active r0 green-fe debug
set platform software trace mdt-pubd chassis active r0 dbal debug
set platform software trace mdt-pubd chassis active r0 tdllib debug
set platform software trace ios chassis active r0 green-be debug
set platform software trace ios chassis active r0 dbal debug
set platform software trace ios chassis active r0 tdllib debug
```

2. Recreate the problem.

3. Collect debug logs:

```
request platform software trace archive last <days>
```

4. Disable debugging using the following commands:

```
set platform software trace mdt-pubd chassis active r0 mdt-ctrl notice
set platform software trace mdt-pubd chassis active r0 pubd notice
set platform software trace mdt-pubd chassis active r0 green-be notice
set platform software trace mdt-pubd chassis active r0 green-fe notice
set platform software trace mdt-pubd chassis active r0 dbal notice
set platform software trace mdt-pubd chassis active r0 tdllib notice
set platform software trace ios chassis active r0 green-be notice
set platform software trace ios chassis active r0 dbal notice
set platform software trace ios chassis active r0 tdllib notice
```

General Telemetry Diagnostics

To capture general telemetry diagnostics, use the following command:

```
show telemetry internal diagnostics
```

Generate a Core

Generate a core using the following commands:

1. `show clock`
2. `configure terminal`
3. `service internal`
4. `end`
5. `request platform software process core mdt-pubd chassis active r0`

Disable Logging

Disable the logging using the following commands:

1. `configure terminal`
2. `no service internal`
3. `end`

Capture CPU Memory

To capture CPU memory details use the following commands:

- `show processes cpu platform sorted | i pubd`
- `show processes memory platform sorted | s pubd`

Cisco Catalyst Center Client Event and SSID Telemetry Filter

Feature History for Cisco Catalyst Center Client Event and SSID Telemetry Filter

This table provides release and related information for the feature explained in this module.

This feature is also available in all the releases subsequent to the one in which they are introduced in, unless noted otherwise.

Table 8: Feature History for Cisco Catalyst Center Client Event and SSID Telemetry Filter

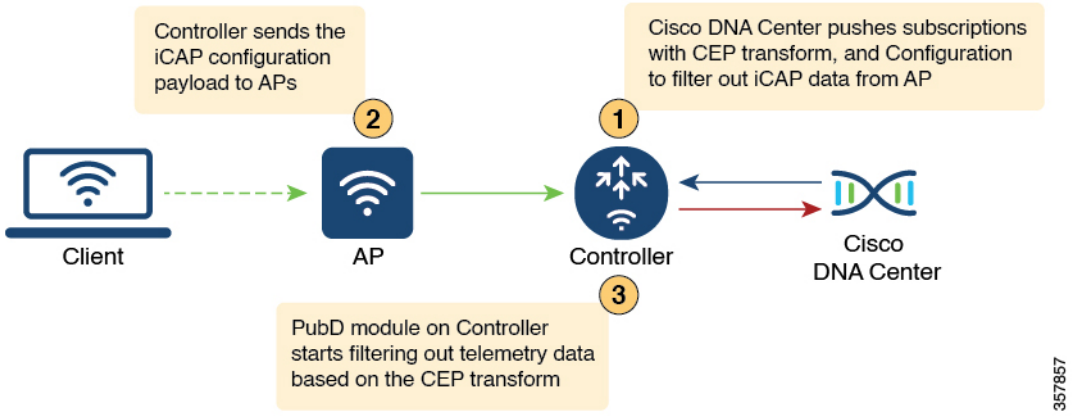
Release	Feature	Feature Information
Cisco IOS XE Dublin 17.10.1	Cisco Catalyst Center Client Event and SSID Telemetry Filter	This feature filters out telemetry data for a configured SSID on the controller and AP.

Information About Cisco Catalyst Center Client Event and SSID Telemetry Filter

Locations such as airports, shopping malls, and so on have wireless guest networks with thousands of transient guest clients. The transient guest clients mix the telemetry data and its subsequent health scores with clients that require assurance (for instance, in a corporate WLAN). This poses a scaling challenge as Cisco Catalyst Center tries to keep up with the receiving high-frequency telemetry data and maintaining history of the transient clients.

This feature addresses the requirement by filtering out the telemetry data for a configured SSID on the controller and AP.

Figure 2: High-Level End-to-End System Flow for Cisco Catalyst Center Client Event and SSID Telemetry Filter



Cisco Catalyst Center configures the Complex Event Processing (CEP) transform with the SSID for which the telemetry data needs to be filtered out along with the subscriptions. The Publishing Daemon (PubD) module in the controller filters out the data based on the configured transform.

Note The Cisco Catalyst Center automation takes care of pushing the transforms. You must enable or disable filtering for a specific SSID in the controller GUI.

To debug the filtering done at PubD, run the following commands in the controller:

```
Device# set platform software trace mdt-pubd chassis active r0 pubd debug
set platform software trace mdt-pubd chassis active r0 mdt-xfrm debug
```

Cisco Catalyst Center configures WLAN for which iCAP data needs to be filtered in an AP profile. The controller then pushes the configuration to the corresponding APs. The AP then programs the **aptrace** module to drop the packets and events for the filtered SSID. The filtered data covers the following:

- Client events
- Client statistics
- AP or RF statistics
- Partial PCAP
- Anomaly detection

Restrictions for Cisco Catalyst Center Client Event and SSID Telemetry Filter

- CLI configuration is applicable for WLAN and not SSID. The Cisco Catalyst Center automation covers one-to-one mapping of WLAN to SSID.
- Controller does not send any notification to Cisco Catalyst Center at the beginning or at the end of filtering.
- Controller GUI configuration is not supported.

Supported Workflow for Cisco Catalyst Center Client Event and SSID Telemetry Filter

- Creating WLANs.
- Mapping WLAN to a Policy Profile.
- Creating a filter for WLAN in AP Join Profile.

Enabling iCAP Filtering in APs (CLI)

Procedure

	Command or Action	Purpose
Step 1	**configure terminal** **Example:** `Device# configure terminal`	Enters global configuration mode.
Step 2	**ap profile** *ap-profile* **Example:** `Device(config)# ap profile xyz-ap-profile`	Configures an AP profile and enters AP profile configuration mode.
Step 3	**description** *ap-profile-name* **Example:** `Device(config-ap-profile)# description "xyz ap profile"`	Adds a description for the AP profile.
Step 4	**icap subscription client exclude telemetry-data wlan** *wlan-profile-name* **Example:** `Device(config-ap-profile)# icap subscription client exclude telemetry-data wlan wlan-name`	Enables iCAP filtering in APs.

Disabling Client Telemetry Data for a WLAN (YANG)

To disable the client telemetry data for a WLAN, use the following RPC model:

```
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
message-id="urn:uuid:0a77124f-c563-469d-bd21-cc625a9691cc">
<nc:edit-config>
<nc:target>
<nc:running/>
</nc:target>
<nc:config>
<site-cfg-data xmlns="http://cisco.com/ns/yang/Cisco-IOS-XE-wireless-site-cfg">
<ap-cfg-profiles>
<ap-cfg-profile>
<profile-name nc:operation="merge">default-ap-profile</profile-name>
<icap-client-exclude-cfgs>
<icap-cient-exclude-cfg nc:operation="merge">
<wlan-profile nc:operation="merge">tel</wlan-profile>
</icap-client-exclude-cfg>
</cap-client-exclude-cfgs>
</ap-cfg-profile>
</ap-cfg-profiles>
</site-cfg-data>
</nc:config>
</nc:edit-config>
</nc:rpc>
```

For more information on YANG models, see the *Cisco IOS XE Programmability Configuration Guide* and YANG Data Models on Github at https://github.com/YangModels/yang/tree/master/vendor/cisco/xe.

You can contact the Developer Support Community for NETCONF/YANG features using the following link:

https://developer.cisco.com/

Verifying Client Telemetry Data for a WLAN

To verify the client telemetry data for a WLAN, use the following command:

```
Device# show running-config | section profile
ap profile default-ap-profile
 capwap retransmit count 8
 capwap timers primary-discovery-timeout 3000
 country IN
 description "default ap profile"
 icap subscription client exclude telemetry-data wlan guest
```