# Enhancements to Streaming Telemetry

This section provides an overview of the enhancements made to streaming telemetry data.



# Hardware Timestamp

***Table 1: Feature History Table***

| Feature Name | Release Information | Description |
|---|---|---|
| Hardware Timestamp | Release 7.3.1 | Whenever periodic statistics are streamed, the collector reads the data from its internal cache, instead of fetching the data from the hardware.<br><br>When the data is read from the cache, the rate at which data is processed shows spikes because the timestamp from the collector is off by several seconds. With hardware timestamping, the inconsistencies that are observed when reading data from the cache file is removed. |

Whenever periodic stats are streamed, the collector reads the stats from its internal cache, instead of fetching the stats from the hardware. When the data is read from the sensor paths of Stats manager cache, the rate calculation shows spikes. This behavior is due to the timestamp from the collector that is off by several seconds. Therefore, timestamp of some other collector takes precedence because timestamps of collectors are not in synchronization with the current timestamp. This is observed when there are multiple collectors providing stats updates for the same interface.

The YANG data model for Stats manager `Cisco-IOS-XR-infra-statsd-oper.yang` is enhanced to enable the collector to read periodic stats data from the router using hardware timestamp.

The hardware timestamp is taken into account when a primary collector (for generic or proto stats) provides stats updates from the hardware to the Stats manager. With hardware timestamping in rate computation while streaming periodic stats, the spikes due to the timestamp issue is resolved.

The hardware timestamp is updated only when the collector attempts to read the counters from hardware. Else, the value remains `0`. The latest stats can be streamed at a minimum cadence of 10 seconds and periodic stats at a cadence of 30 seconds. The support is available only for physical interfaces and subinterfaces, and bundle interface and subinterfaces.

When there is no traffic flow on protocols for an interface, the hardware timestamp for the protocols is published as 0. This is due to non-synchronized timestamps sent by the collector for protocols in traffic as compared to non-traffic scenarios.

A non-zero value is published for protocols that have stats published by a primary collector for both traffic and non-traffic scenarios.

**Note** The hardware timestamp is supported only for primary collectors. When the hardware has no update, the timestamp will be same. However generic counters are computed for primary and non-primary collectors. The non-primary collectors show the latest stats, but not the timestamp.

When the counters are cleared for an interface using **clear counters interface** command, all counter-related data including the timestamps for the interface is cleared. After all counter values are cleared and set to `0`, the last data time is updated only when there is a request for it from a collector. For example, last data time gets updated from a collector:

```
Router#:Aug  7 09:01:08.471 UTC: statsd_manager_l[168]: Updated last data time for ifhandle
 0x02000408,
stats type 2 from collector with node 0x100, JID 250, last data time 1596790868.
INPUT: last 4294967295 updated 1596469986. OUTPUT: last 4294967295 updated 1596469986
```

All other counter values and hardware timestamp are updated when the counters are fetched from the hardware. In this case, all counters including the hardware timestamp is 0:

```
{"node_id_str":"MGBL_MTB_5504","subscription_id_str":"app_TEST_200000001",
"encoding_path":"Cisco-IOS-XR-infra-statsd-oper:infra-statistics/interfaces/interface/cache/generic-counters",
"collection_id":"7848",
"collection_start_time":"1596790879567",
"msg_timestamp":"1596790879571","data_json":
[{"timestamp":"1596790879570","keys":[{"interface-name":"FortyGigE0/1/0/11"}],
"content":{"packets-received":"0","bytes-received":"0","packets-sent":"0",
"bytes-sent":"0","multicast-packets-received":"0","broadcast-packets-received":"0",
"multicast-packets-sent":"0","broadcast-packets-sent":"0","output-drops":0,"output-queue-drops":0,
"input-drops":0,"input-queue-drops":0,"runt-packets-received":0,"giant-packets-received":0,
"throttled-packets-received":0,"parity-packets-received":0,"unknown-protocol-packets-received":0,
"input-errors":0,"crc-errors":0,"input-overruns":0,"framing-errors-received":0,"input-ignored-packets":0,
"input-aborts":0,"output-errors":0,"output-underruns":0,"output-buffer-failures":0,"output-buffers-swapped-out":0,
"applique":0,"resets":0,"carrier-transitions":0,"availability-flag":0,
"last-data-time":"1596790868","hardware-timestamp":"0",
"seconds-since-last-clear-counters":15,"last-discontinuity-time":1596469946,"seconds-since-packet-received":0,
"seconds-since-packet-sent":0}}],"collection_end_time":"1596790879571"}
```

# Monitor Process State via Event-Driven Telemetry

*Table 2: Feature History Table*

| Feature Name | Release Information | Description |
|---|---|---|
| Monitor Process State via Event-Driven Telemetry (EDT) | Release 7.4.2 | With this feature, you can configure the list of processes you want to monitor, and receive notifications via event-driven telemetry when the configured process restarts or crashes.<br>This feature introduces the **process-state-monitor location** command to monitor processes on specific nodes or all nodes. |

You can monitor the state of Cisco IOS XR processes using event-driven telemetry (EDT) notifications.

You can use the **process-state-monitor location** [*node* | **all**] command or **Cisco-IOS-XR-wd-proc-state-cfg.yang** data model to perform the following operations:

- Configure the list of process names to register for EDT notifications for a specific node or for all nodes.
- Receive notification via EDT about the end of the configured processes.

**Step 1** Register the processes that you want to monitor and receive notifications:

- Configure the process for a specific node.

**Configure via CLI:**

```
Router(config)#process-state-monitor location 0/RP1/CPU0
Router(config-set-proc-name)#process-name l2vpn_mgr
Router(config-set-proc-name)#process-name ipv4_rib
Router(config-set-proc-name)#process-name ipv6_rib
Router(config-set-proc-name)#exit
Router(config)#process-state-monitor location 0/1/CPU0
Router(config-set-proc-name)#process-name l2fib_mgr
Router(config-set-proc-name)#commit
```

**Configure via Data Model:**

```
<process-state-monitor-node-specific
xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-wd-proc-state-cfg">
 <node>
  <node>0/0/CPU0</node>
   <process-state-monitor>
    <process-names>
     <process-name>
      <process-name>l2fib_mgr</process-name>
     </process-name>
    </process-names>
   </process-state-monitor>
  </node>
 <node>
  <node>0/RP0/CPU0</node>
   <process-state-monitor>
    <process-names>
     <process-name>
```

```
      <process-name>l2vpn_mgr</process-name>
      </process-name>
     <process-name>
    <process-name>ipv4_rib</process-name>
   </process-name>
    <process-name>
     <process-name>ipv6_rib</process-name>
   </process-name>
  </process-names>
 </process-state-monitor>
</node>
</process-state-monitor-node-specific>
```

**Note** The configuration that you apply to active RP is not synchronized with the standby RP. So on switch over, the processes that are monitored on old active RP are not monitored on the new active RP.

If a process is configured under a specific location (such as RP0, LC0, LC1 and so on), and if you configure the same process under the same node, then the configuration is applied successfully. However, the process will not be registered again with the process manager to avoid receiving duplicate notifications.

- Configure the process on all the nodes.

  **Configure via CLI:**

  ```
  Router(config)#process-state-monitor location all
  Router(config-set-proc-name)#process-name dumper
  Router(config-set-proc-name)#commit
  ```

  **Configure via Data Model:**

  ```
  <process-state-monitor xmlns="http://cisco.com/ns/yang/Cisco-IOS-XR-wd-proc-state-cfg">
   <location>
    <all>
     <process-names>
      <process-name>
       <process-name>dumper</process-name>
      </process-name>
     </process-names>
    </all>
   </location>
  </process-state-monitor>
  ```

  **Note** In addition to the active RP, if the configuration must be applied to the standby RP, then provide the location of standby RP when configuring the process. If the configuration is applicable to all nodes (RP, LC) then select the location as `all`.

  If the process is configured for all locations, and if you configure the same process under some other location (such as RP0, LC0, LC1 and so on), then the commit operation fails with an error message. When a process on any one of the nodes has already reached the maximum limit, the configuration is not applied on that node.

To remove the configuration, use the **no** form of the command. For example, the process to monitor the L2VPN manager is removed from the configuration:

```
Router(config-monitor-proc-name)#no process-name l2vpn_mgr
```

**Step 2** View the configuration applied on the router.

**Example:**

```
Router#show running-config
Thu Feb 3 06:12:00.014 UTC
Building configuration...
!! IOS XR Configuration 7.4.2
```

```
!! Last configuration change at Thu Feb 3 06:11:50 2022 by cisco
!
username cisco
group root-lr
group cisco-support!
call-home
service active
contact smart-licensing
profile Test
active
destination transport-method email disable
destination transport-method http
!
!
process-state-monitor location all
 process-name dumper
!
process-state-monitor location 0/0/CPU0
 process-name l2fib_mgr
!
process-state-monitor location 0/RP0/CPU0
 process-name l2vpn_mgr
!
```

The following is a sample EDT notification that shows the data that is collected when the process restarts or crashes:

```
node_id_str: "ios"
subscription_id_str: "proc-state"
encoding_path: "Cisco-IOS-XR-wd-proc-state-oper:process-death-notification/process-death-info"
model_version: "2019-04-05"
collection_id: 1
collection_start_time: 1623890312688
msg_timestamp: 1623890312688
data_gpbkv {
  timestamp: 1623890312680
  fields {
    name: "content"
    fields {
      name: "location"
      string_value: "0_RP0_CPU0"
    }
    fields {
      name: "process-name"
      string_value: "l2vpn_mgr"
    }
    fields {
      name: "pid"
      uint32_value: 9743
    }
    fields {
      name: "jid"
      uint32_value: 1182
    }
  }
}
```

# Stream Telemetry Data about PBR Decapsulation Statistics

You can stream telemetry data about PBR decapsulation statistics for GRE and GUE encapsulation protocols that deliver packets using IPv4 or IPv6. The encapsulated data has source and destination address that must match with the source and destination address in the classmap. Both encapsulation and decapsulation interfaces collect statistics periodically. The statistics can be displayed on demand using **show policy-map type pbr [vrf vrf-name] address-family ipv4/ipv6 statistics** command. For more information on PBR-based decapsulation, see *Interface and Hardware Component Configuration Guide for Cisco ASR 9000 Series Routers*.

With this release, the decapsulation statistics can be displayed using `Cisco-IOS-XR-infra-policymgr-oper.yang` data model and telemetry data. You can stream telemetry data from the sensor path:

`Cisco-IOS-XR-infra-policymgr-oper:policy-manager/global/policy-map/policy-map-types/policy-map-type/vrf-table/vrf/afi-table/afi/stats`

The following steps show the PBR configuration and the decapsulation statistics that is streamed as telemetry data to the collector.

**Step 1** Check the running configuration to view the configured PBR per VRF.

**Example:**

```
Router#show running-config
Building configuration...
!! IOS XR Configuration 0.0.0
!!
vrf vrf1
 address-family ipv4 unicast
 !
 address-family ipv6 multicast
 !
!
netconf-yang agent
 ssh
!
!
class-map type traffic match-all cmap1
 match protocol gre
 match source-address ipv4 161.0.1.1 255.255.255.255
 match destination-address ipv4 161.2.1.1 255.255.255.255
 end-class-map
!
policy-map type pbr gre-policy
 class type traffic cmap1
  decapsulate gre
 !
 class type traffic class-default
 !
 end-policy-map
!
interface GigabitEthernet0/0/0/1
 vrf vrf1
 ipv4 address 2.2.2.2 255.255.255.0
 shutdown
!
vrf-policy
 vrf vrf1 address-family ipv4 policy type pbr input gre-policy
!
```

```
end
```

**Step 2** View the output of the VRF statistics.

**Example:**

```
Router#show policy-map type pbr vrf vrf1 addr-family ipv4 statistics

VRF Name:       vrf1
Policy-Name:    gre-policy
Policy Type:    pbr
Addr Family:    IPv4

Class:     cmap1
     Classification statistics      (packets/bytes)
       Matched             :       13387587/1713611136
     Transmitted statistics         (packets/bytes)
       Total Transmitted   :       13387587/1713611136

Class:     class-default
     Classification statistics      (packets/bytes)
       Matched             :              0/0
     Transmitted statistics         (packets/bytes)
       Total Transmitted   :              0/0
```

After you have verified that the statistics are displayed correctly, stream telemetry data and check the streamed data at the collector. For more information about collectors, see *Operate on Telemetry Data for In-depth Analysis of the Network* section in the Monitor CPU Utilization Using Telemetry Data to Plan Network Infrastructure chapter.

```
ios.0/0/CPU0/ $ mdt_exec -s Cisco-IOS-XR-infra-policymgr-oper:policy-manager
/global/policy-map/policy-map-types/policy-map-type/vrf-table/vrf/afi-table/afi/stats -c 100
{"node_id_str":"ios","subscription_id_str":"app_TEST_200000001","encoding_path":
"Cisco-IOS-XR-infra-policymgr-oper:policy-manager/global/policy-map/policy-map-types/policy-map-type
/vrf-table/vrf/afi-table/afi/stats","collection_id":"1","collection_start_time":"1601361558157",
"msg_timestamp":"1601361559179","data_json":[{"timestamp":"1601361559178","keys":[{"type":"ipv6"},
{"vrf-name":"vrf_gue_ipv4"},{"type":"ipv4"}],"content":{"pmap-name":"gre-policy","vrf-name":
"vrf1","appln-type":2,"addr-family":1,"rc":0,"plmgr-vrf-stats":[{"pmap-name":"gre-policy",
"cmap-stats-arr":[{"cmap-name":"cmap1","matched-bytes":"1713611136","matched-packets":"13387587",
"transmit-bytes":"1713611136","transmit-packets":"13387587"}]}]}}],
"collection_end_time":"1601361559183"}
---------------------------------  snipped for brevity ------------------------------------
```