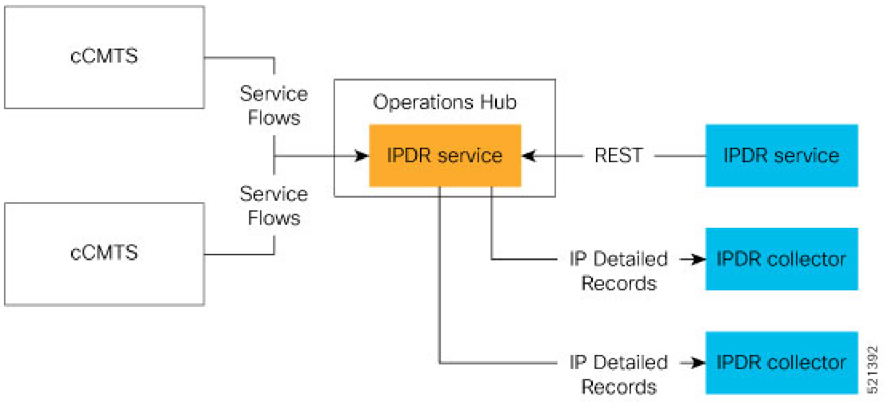
External Interfaces Support for Cisco Cloud Native Broadband Router

Cisco cnBR supports legacy interface translation, general network management, and monitoring information.

Cisco cnBR supports the following external interfaces:

IP Detail Record Service

The Cisco Operations Hub hosted IP Detail Record (IPDR) service provides the mechanisms to export IP detailed records to IPDR collectors and the ability to configure the IPDR service.



The Cisco Operations Hub IPDR service operates in a similar way as other Cisco Cable Modem Termination Systems (CMTS) products. You can configure it through the REST interface. See IPDR Streaming Protocol on the Cisco CMTS Routers for reference.

The URL `https://{Hostname}/api/ipdr` is created for the IPDR service, which is used for the REST configuration and status requests. The collector connects to the IPDR service on default port 4737 to establish a TCP session. Then, IPDR records are streamed from the IPDR service to the collector over this TCP session.

For the IPDR service to deliver records, the IP address of the collector that receives the records is required. An ordered list of collectors is contained in a session. Only one collector in a session receives the records, the others are available as backup. The session describes the delivery mechanism and record format. You can define multiple sessions so that more than one collector can receive IPDR records from Operations Hub.

Terminology

Term	Description
Collector	The host that receives (collects) the IPDR records.
Exporter	The IPDR service includes an exporter service that generates the IPDR records.
Session	Describes the set of collectors and templates that are used to send IPDR records. At a time, only one collector in a session gets IPDR data at a time based on a priority order. If a collector is unavailable, the collector with the next highest priority gets the records.
Template	Identifies the record format for sending the records.

Configure IPDR Service

To configure the IPDR service, use a single command to set all configuration parameters in JSON format in one single action. This configuration method overwrites the existing configuration and activates the new configuration.

```
/v1/config
```

Note `/ipdr/config` is deprecated but usable.

To set the configuration, use the **PUT** HTTP method as shown in the following example.

```
curl -k -X PUT -H "Content-Type: application/json" -d @- << EOF
https://{Hostname}/api/ipdr/v1/config
{json_string}
EOF
```

Note Parameter -k allows insecure server connections when using SSL.

Example: Add or change IPDR configuration.

```
curl -k -X PUT -H "Content-Type: application/json" -d @- << EOF
https://opshub1.cisco.com/api/ipdr/v1/config
{
    "sessions": [
        {
            "id": 1,
            "name": "session_1",
            "description": "IPDR Session 1",
            "type": {
                "type": "time-interval",
                "interval": 15
            },
            "templates": [
                {
                    "template-type": "SAMIS-TYPE1"
            }
        ],
        "associated-collectors": [
            {
                "collector-name": "Collector1",
                "priority": 1
            }
        ]
    },
    {
        "id": 2,
        "name": "session_2",
        "description": "IPDR Session 2",
        "type": {
            "type": "event",
            "interval": 0
        },
        "templates": [
            {
                "template-type": "DS-UTIL"
            }
        ],
        "associated-collectors": [
            {
                "collector-name": "Collector1",
                "priority": 1
            }
        ]
    },
    {
        "id": 3,
        "name": "session_3",
        "description": "IPDR Session 3",
        "type": {
            "type": "event",
            "interval": 0
        },
        "templates": [
            {
                "template-type": "US-UTIL"
            }
        ],
        "associated-collectors": [
            {
                "collector-name": "Collector1",
                "priority": 1
            }
        ]
    }
],
"collectors": [
    {
        "name": "Collector1",
        "address": "10.0.0.1",
        "nat-address": "0.0.0.0",
        "port": 0
    }
],
"exporter": {
    "ack-timeout": 60,
    "keep-alive": 300,
    "max-unacked": 200,
    "started": true
},
    "utilization": {
        "interval": 300
    }
}
EOF
```

After setting the configuration, you can use the **GET** HTTP method as shown in the following example to display the consolidated configuration:

```
curl -H 'Content-Type: application/json' -X GET https://{Hostname}/api/ipdr/v1/config
```

Example: Display the existing IPDR configuration

```
curl -k -H 'Content-Type: application/json' -X GET
https://opshub1.cisco.com/api/ipdr/v1/config
{
    "sessions": [
        {
            "id": 1,
            "name": "session_1",
            "description": "IPDR Session 1",
            "type": {
                "type": "time-interval",
                "interval": 2
            },
            "templates": [
                {
                    "template-type": "SAMIS-TYPE1"
                }
            ],
            "associated-collectors": [
                {
                    "collector-name": "Collector1",
                    "priority": 1
                }
            ]
        },
        {
            "id": 2,
            "name": "session_2",
            "description": "IPDR Session 2",
            "type": {
                "type": "event",
                "interval": 0
            },
            "templates": [
                {
                    "template-type": "DS-UTIL"
                }
            ],
            "associated-collectors": [
                {
                    "collector-name": "Collector1",
                    "priority": 1
                }
            ]
        },
        {
            "id": 3,
            "name": "session_3",
            "description": "IPDR Session 3",
            "type": {
                "type": "event",
                "interval": 0
            },
                "templates": [
                    {
                        "template-type": "US-UTIL"
                    }
                ],
                "associated-collectors": [
                    {
                        "collector-name": "Collector1",
                        "priority": 1
                    }
                ]
            }
        ],
        "collectors": [
            {
                "name": "Collector1",
                "address": "10.0.0.1",
                "nat-address": "0.0.0.0",
                "port": 0
            }
        ],
        "exporter": {
            "ack-timeout": 60,
            "keep-alive": 300,
            "max-unacked": 200,
            "started": true
        },
        "utilization": {
            "interval": 240
        }
}
```

Example: Remove IPDR configuration

```
curl -X PUT -H "Content-Type: application/json" https://opshub1.cisco.com/api/ipdr/v1/config
```

Note The `opshub1.cisco.com` is only for illustrative purposes. Use the Fully Qualified Domain Name (FQDN) of the Cisco Operations Hub deployed at your site.

Fields In JSON

This table lists the fields used in JSON and their description.

Field Name	Description	Type	Enforcement
ack-timeout	Exporter timeout, after which an acknowledgement is received from the collector before retry.	Number. 5–60 seconds; the default value is 60.	Optional
address	The IP address of the collector, which is used to receive the IPDR records.	IP Address	Required
collector-name	A specific collector definition for collectors.	String	Required
collectors:name	Unique name used to identify a collector.	String	Required
description	Long descriptive text.	String	Required

Field Name	Description	Type	Enforcement
id	A unique session number for the purpose of reference.	Number	Required
interval	The interval used to send DS-UTIL and US-UTIL data.	Number in seconds, **0** means disabled.	Optional
keep-alive	The keepalive time after which the collector is considered unavailable.	Number. 5–300 seconds; the default value is 300.	Optional
max-unacked	The maximum number of unacknowledged records.	Number. 5–200; the default value is 200.	Optional
name	Descriptive name for reference purposes.	String	Required
nat-address	The NAT IP address of the collector.	IP Address	Optional
port	The port of the collector.	Number	Optional
priority	The order to use the collector. Use the collector with the lowest priority number first.	> 0	Required
started	Start the IPDR service or not.	Boolean	Required
type:type	The method used to request data from the service.	String. Possible values: adhoc, event, time-interval	Required
type:interval	The frequency of sending the data for a session.	2–1440 minutes.	Required only if "type:type" field is set to "time-interval".
template-type	Identifies the records format.	String. Possible values: SAMIS-TYPE1, US-UTIL, DS-UTIL, TOPOLOGY	Required

REST Return Codes

You can use the status codes listed in the following table to convey the results of a request.

Code	Short Description	Response Text	Actions
400	HTTP_BAD_REQUEST	• Failure: request format error. • Failed to configure session when exporter starts, stop it at first.	Confirm that the format of the request is valid or restart the IPDR service to apply new sessions.
404	HTTP_NOT_FOUND	• Failure: collector doesn't exist.	Return this code when adding a session referring to a collector that does not exist. If it is a consolidated configuration request, correct the request to include a valid collector.

Code	Short Description	Response Text	Actions
500	HTTP_BAD_REQUEST	• Failed to add new session to cache. • Failed to apply IPDR configuration. • Failed to config ipdr session to exporter. • Failed to get ipdr sessions with internal error. • Failed to recover configurations. • Failed to remove session in cache. • Failed to revert session in cache when db failed. • Failed to update IPDR configuration. • Failure: allocate JSON object error. • Failure: get ipdr config information error. • Failure: not enough memory. • Failure: save global cfg error. • IPDR configuration not updated, restored to original.	Internal error that requires engineering team engagement.
503	HTTP_SER_UNAVAIL	• not ready	Use this code only in response to readiness check. If the service is not ready, confirm that the Cassandra database is ready. Otherwise, get the database ready. If the Cassandra database is ready and operational, ask for customer support.

Monitor

Use the **GET** HTTP method of the following REST APIs to monitor the status of the IPDR session, collector, and exporter.

Monitor Session Status

- Get the status of all sessions.

  ```
  /v1/sessions
  ```

- Get the status of a specific session.

  ```
  /v1/sessions/{id}
  ```

Note `/ipdr/session/status` is deprecated but usable.

Example:

```
curl -k -H 'Content-Type: application/json' -X GET
https://opshub1.cisco.com/api/ipdr/v1/sessions
Session ID: 1, Name: samis, Descr: samis, Started: True
Session Type: Time Interval (15 minutes).
Expires in 81 seconds.
Exporting not started.
2019-05-29T05:08:14 Statistics:
 Transmitted 0 Acknowledged 0 Enqueued 0 Lost 0
 queuedOutstanding 0 queuedUnacknowledged 0
1 Collectors in the session:
 Name: collector1, IPAddr: 10.0.0.1, Port: N/A, Priority: 1[DISCONNECTED]
Templates in the session:
 Template ID: 2, Name:
http://www.cablelabs.com/namespaces/DOCSIS/3.0/xsd/ipdr/DOCSIS-SAMIS-TYPE-1/DOCSIS-SAMIS-TYPE-1_3.5.1-A.1.xsd,
 Type: SAMIS-TYPE-1, KeyNumber: 28
Session 1 has a total of 1 templates.
Session ID: 2, Name: cmts-ds-util-stats, Descr: cmts-ds-util-stats, Started: True
Session Type: Event Based.
2019-05-29T05:08:14 Statistics:
 Transmitted 0 Acknowledged 0 Enqueued 0 Lost 0
 queuedOutstanding 0 queuedUnacknowledged 0
1 Collectors in the session:
 Name: collector1, IPAddr: 10.0.0.1, Port: N/A, Priority: 0[DISCONNECTED]
Templates in the session:
 Template ID: 13, Name:
http://www.cablelabs.com/namespaces/DOCSIS/3.0/xsd/ipdr/DOCSIS-CMTS-DS-UTIL-STATS-TYPE/DOCSIS-CMTS-DS-UTIL-STATS-TYPE_3.5.1-A.3.xsd,
 Type:
http://www.cablelabs.com/namespaces/DOCSIS/3.0/xsd/ipdr/DOCSIS-CMTS-DS-UTIL-STATS-TYPE/DOCSIS-CMTS-DS-UTIL-STATS-TYPE_3.5.1-A.3.xsd,
 KeyNumber: 11
Session 2 has a total of 1 templates.
Session ID: 3, Name: cm-status, Descr: cm-status, Started: True
Session Type: Ad-hoc.
Exporting not started.
2019-05-29T05:08:14 Statistics:
 Transmitted 0 Acknowledged 0 Enqueued 0 Lost 0
 queuedOutstanding 0 queuedUnacknowledged 0
1 Collectors in the session:
 Name: collector1, IPAddr: 10.0.0.1, Port: N/A, Priority: 1[DISCONNECTED]
Templates in the session:
 Template ID: 8, Name:
http://www.cablelabs.com/namespaces/DOCSIS/3.0/xsd/ipdr/DOCSIS-CMTS-CM-REG-STATUS-TYPE/DOCSIS-CMTS-CM-REG-STATUS-TYPE_3.5.1-A.1.xsd,
 Type:
http://www.cablelabs.com/namespaces/DOCSIS/3.0/xsd/ipdr/DOCSIS-CMTS-CM-REG-STATUS-TYPE/DOCSIS-CMTS-CM-REG-STATUS-TYPE_3.5.1-A.1.xsd,
 KeyNumber: 18
Session 3 has a total of 1 templates.
```

```
curl -k -H 'Content-Type: application/json' -X GET
https://opshub1.cisco.com/api/ipdr/v1/sessions/1
Session ID: 1, Name: samis, Descr: samis, Started: True
Session Type: Time Interval (15 minutes).
Expires in 81 seconds.
Exporting not started.
2019-05-29T05:08:14 Statistics:
 Transmitted 0 Acknowledged 0 Enqueued 0 Lost 0
 queuedOutstanding 0 queuedUnacknowledged 0
1 Collectors in the session:
 Name: collector1, IPAddr: 10.0.0.1, Port: N/A, Priority: 1[DISCONNECTED]
```

```
Templates in the session:
 Template ID: 2, Name:
http://www.cablelabs.com/namespaces/DOCSIS/3.0/xsd/ipdr/DOCSIS-SAMIS-TYPE-1/DOCSIS-SAMIS-TYPE-1_3.5.1-A.1.xsd,
 Type: SAMIS-TYPE-1, KeyNumber: 28
Session 1 has a total of 1 templates.
```

Monitor Collector Status

`/v1/collectors`

Note `/ipdr/collectors/status` is deprecated but usable.

Example:

```
curl -k -H 'Content-Type: application/json' -X GET
https://opshub1.cisco.com/api/ipdr/v1/collectors
Collector name collector1, ip addr 10.0.0.1, port 0
```

Monitor Exporter Status

`/v1/exporter`

Note `/ipdr/exporter/status` is deprecated but usable.

Example:

```
curl -k -H 'Content-Type: application/json' -X GET
https://opshub1.cisco.com/api/ipdr/v1/exporter
IPDR exporter is started.
Current parameters:
    KeepAliveInterval:  300
      AckTimeInterval:   60
  AckSequenceInterval:  200
```

Simple Network Management Protocol

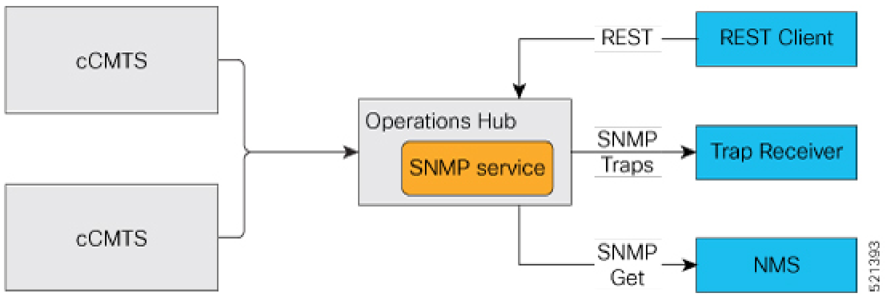
The Simple Network Management Protocol (SNMP) allows you to monitor the DOCSIS elements of Cisco cnBR.

The REST API is the recommended method to configure and operate Cisco cnBR. However, partial SNMP functionality is provided for compatibility with legacy SNMP applications. The Cisco cnBR SNMP Agent is located on the Cisco Operations Hub, and not on individual Cisco cnBRs.

SNMP aggregates information from multiple Cisco cnBR cores that are managed by Cisco Operations Hub.

From an application perspective, you must consider the Cisco Operations Hub as a *large* Cisco cnBR.

The following image provides you an overview of how the SNMP works in the Cisco cnBR.



Configure SNMP

Follow these steps to configure SNMP for Cisco cnBR:

Use the REST API to configure the SNMPv2 community string or Trap Receivers.

```
curl -X {GET|PUT|DELETE} https://{hostname}/api/snmp/v1/config
```

Use one of the following options:

- **SNMPv2 Community**

  To configure SNMPv2 Community, replace `<opshub-ip>` with the Cisco Operations Hub IP. The following example is only indicative. See the *Cisco Cloud Native Broadband Router Operations Hub REST API Guide* for the authentication and encryption format.

```
curl -X GET https://{hostname}/api/snmp/v1/config
{"community-list":[],"v3user-list":[],"trap-receivers":[],"trap-enabled-list":[]}

curl -X PUT -d @- << EOF https://{hostname}/api/snmp/v1/config
{
  "community-list": [
    {
      "community": "public",
      "access": "ro",
      "source": "",
      "oid": ""
    }
  ]
}
EOF

curl -X GET https://{hostname}/api/snmp/v1/config
{"community-list":[{"community":"public","access":"ro","source":"","oid":""}],
"v3user-list":[],"trap-receivers":[],"trap-enabled-list":[]}

curl -X DELETE -d @- << EOF https://{hostname}/api/snmp/v1/config
{
  "community-list": [
    {
      "community": "public",
      "access": "ro",
      "source": "",
      "oid": ""
    }
  ]
}
EOF
```

- **SNMPv1/v2 Trap**

  The Trap Receiver is a server listening to a specific UDP port for SNMP Trap events. Use the following REST API to configure the Trap Receiver IP address, port, and other information in the Cisco cnBR SNMP agent. The REST API enables the agent to send traps to the trap receiver.

```
curl -X PUT -d @- << EOF https://{hostname}/api/snmp/v1/config
{
  "trap-receivers": [
    {
      "host": "10.1.1.2",
      "port": 12348,
      "version": 2,
      "community": "private"
    }
  ]
}
EOF

curl -X GET https://{hostname}/api/snmp/v1/config
{"community-list":[],"v3user-list":[],"trap-receivers":
[{"host":"1.1.2.2","port":12345,"version":1,"community":"public"},
{"host":"10.1.1.2","port":12348,"version":2,"community":"private"}]}

curl -X DELETE -d @- << EOF https://{hostname}/api/snmp/v1/config
{
  "trap-receivers": [
    {
      "host": "1.1.2.2"
    }
  ]
}
EOF
```

Note

- `host`: Trap Receiver's IP address. For DELETE action, *host* is the key, and the other fields are not necessary.
- `port`: Trap Receiver listens on this port. The Trap Receiver uses the default port **162**, if the *port* is not specified.
- `version`: 1 for SNMPv1, 2 for SNMPv2.
- `community`: Specify the community string to send or receive trap. At the receiver side, there is a configuration file to specify the *community*.

SNMP Support Scope

MIB

The following tables are supported.

```
docsIf31CmtsDsOfdmChanTable
docsIf31DocsisBaseCapability
docsIf3CmtsCmRegStatusTable
docsIf3CmtsCmUsStatusTable
docsIf3DsChSetTable
```

```
docsIf3MdChCfgTable
docsIf3MdDsSgStatusTable
docsIf3MdNodeStatusTable
docsIf3MdUsSgStatusTable
docsIf3UsChSetTable
docsIfCmtsChannelUtilizationInterval
docsIfCmtsChannelUtilizationTable
docsIfCmtsCmStatusTable
docsIfCmtsDownChannelCounterTable
docsIfCmtsModulationTable
docsIfCmtsUpChannelCounterTable
docsIfDocsisBaseCapability
docsIfDownstreamChannelTable
docsIfUpstreamChannelTable
docsPnmBulkDestIpAddr
docsPnmBulkDestIpAddrType
docsPnmCmtsUtscCfgTable
docsPnmCmtsUtscCtrlTable
docsQos3CmtsMacToSrvFlowTable
docsQos3ServiceFlowStatsTable
docsQos3ServiceFlowTable
docsRphyCmtsCmRegStatusTable
docsRphyRpdDevIdentificationTable
docsRphyRpdDevNdfCfgTable
docsRphyRpdDevNdrCfgTable
docsRphyRpdIfCoreToRpdMapTable
docsRphyRpdIfRpdToCoreMapTable
docsRphyStatsRpdUsOfdmaChanPerfStatsTable
docsRphyStatsRpdUsScQamChanPerfStatsTable
ifTable
```

Note

- Only a subset of OIDs required for the third-party tools integration is supported.
- Only the following MIBs supports SNMP Write:

  ```
  docsPnmBulkDestIpAddr
  docsPnmBulkDestIpAddrType
  docsPnmCmtsUtscCfgTable
  docsPnmCmtsUtscCtrlTable
  ```

- Cisco cnBR does not support NDF/NDR. The following MIBs only conform to prerequisites of third-party tools to capture upstream spectrum:

  ```
  docsRphyRpdDevNdfCfgTable
  docsRphyRpdDevNdrCfgTable
  ```

- For the following MIB, the table returns value zero (0) for all rows until OFDMA is supported by Cisco cnBR.

  ```
  docsRphyStatsRpdUsOfdmaChanPerfStatsTable
  ```

Trap

Only CM online and offline events are supported.

Reference

DOCSIS MIBs

SNMP Limitations

Cisco cnBR SNMP has the following limitations:

- SNMP write is supported only for the MIB object or table that is listed in the MIB section. For more information, see OIDs of MIB tables supported by Cisco Operations Hub.
- Only a limited set of DOCSIS MIB OIDs and traps is supported. For more information, see OIDs of MIB tables supported by Cisco Operations Hub.