

Network Visibility Module Collector Installation and Configuration Guide, Release 5.1.x

Introduction to Network Visibility Module Collector 2

Components Of and Requirements For Network Visibility Module Collector 2

Hardware Sizing for Standalone Network Visibility Module Collector 2

Internet Protocol Flow Information Export (IPFIX) 3

Network Visibility Module Collector Exported Data Fields 3

Host Firewall Recommendation for NVM Collector 9

Set Up Network Visibility Module Collector 9

Set Up Collector DTLS 13

Configure Collector for DTLS 13

Set Up Collector mDTLS 14

Configure Collector for mDTLS 14

Validate Installation of Network Visibility Module 15

Validate Collector Status 15

Collector Diagnostic Tool 15

Basic Troubleshooting 16

Related Documentation for Network Visibility Module Collector 18

Revised: November 18, 2025,

Introduction to Network Visibility Module Collector

This document describes how to install and configure the Network Visbility Module (NVM) Collector that you can download from the Cisco Software Download page. It describes its components, how to set it up, and then how to validate installation and collector status, as well as perform basic troubleshooting. Refer to the Release Notes for Cisco Secure Client Network Visibility Module Collector for additional information.

Cisco Secure Client is a unified agent that delivers multiple security services to protect the enterprise, and it also supports additional modules that cater to different aspects of enterprise security. The additional modules enable security features like posture assessment, malware protection, roaming security, and more.

Cisco Secure Client Network Visibility Module provides a continuous feed of high-value endpoint telemetry. Network Visibility Module allows organizations to see endpoint and user behaviors on their networks. It collects flow from endpoints both on- and off-premise, along with valuable context like users, applications, devices, locations, and destinations. It caches this data and sends it to the Network Visibility Module Collector when it is on a trusted network (the corporate network on-prem or through VPN).

Network Visibility Module Collector is a server that receives Internet Protocol Flow Information Export (IPFIX), on page 3 data and optionally filters, and then exports it to syslog or Splunk HTTP Event Collector (HEC). Network Visibility Module Collector processes received messages that adhere to the nvzFlow protocol specification: https://developer.cisco.com/site/network-visibility-module/. You can install the collector on a standalone Linux system.

Refer to Splunk documentation on how to set up and use HTTP Event Collector.

Components Of and Requirements For Network Visibility Module Collector

Cisco recommends that you have knowledge of these topics:

- Cisco Secure Client with Network Visibility Module
- Cisco Secure Client licensing
- Network Visibility Module Collector licensing

If you run NVM Collector on a separate Linux device, you should plan for 35-40K endpoints per device, using this general scaling:

- CPU/memory sizing can be reduced
- Disk input/output will not be applicable since logging is only being done for the collector and Linux
- 50GB of disk space available to run the OS and collector components

Hardware Sizing for Standalone Network Visibility Module Collector

The following provides recommended hardware requirements for a standalone Network Visibility Module Collector instance running on 64-bit Linux:



Note By default, Network Visibility Module multiprocess mode is enabled.

- Up to 1000 endpoints/server instance:
  - CPU cores: 6 cores / 2.2 GHz (x86 64-bit)
  - RAM size: 8 GB
  - Combined IOPS: 800 input/output operations per second (IOPS)
  - Disk sub-system: Any (minimum 10k RPM)
  - Total disk capacity: 50 GB
- 1000-5000 endpoints/server instance:
  - CPU cores: 8 cores / 2.4 GHz (x86 64-bit)
  - RAM size: 16 GB
  - Combined IOPS: 1000 input/output operations per second (IOPS)
  - Disk sub-system: Any (minimum 10k RPM)
  - Total disk capacity: 50 GB
- 7500-10,000 endpoints/server instance:
  - CPU cores: 12 cores / 2.6 GHz (x86 64-bit)
  - RAM size: 24 GB
  - Combined IOPS: 1200 input/output operations per second (IOPS)
  - Disk sub-system: Any (minimum 10k RPM)
  - Total disk capacity: 50 GB

Internet Protocol Flow Information Export (IPFIX)

IPFIX is an IETF protocol that defines a standard for exporting IP flow information for various purposes such as accounting, auditing, or security. IPFIX is based on Cisco NetFlow protocol v9, although it is not directly compatible. Cisco nvzFlow is a protocol specification based on the IPFIX protocol. By design, IPFIX is an extensible protocol allowing you to define new parameters to convey information. Cisco nvzFlow protocol extends the IPFIX standard and defines new information elements, as well as a standard set of IPFIX templates that are conveyed as part of the telemetry used by Cisco Secure Client Network Visibility Module.

IPFIX flow templates are sent to the collector at the start of the IPFIX communication. These templates help the collector make sense of the IPFIX data. In rare circumstances, a template may not be found. This issue is identified with a **No template found** message in a packet capture on the endpoint or a **No templates for flowset** message in the collector logs. To remedy this issue, restart one of the endpoints.

Network Visibility Module Collector Exported Data Fields

Network Visibility Module Collector exports six kinds of data records: endpoint, interface, flow, process, osquery, and Cisco EVE (Encrypted Visibility Engine). Each data record is a set of field key-value pairs. The table below provides the field keys, field names, and context for the kind of data record in which the given field key is present.

Field Key	Field Name	Field Context	Field Description
agv	AgentVersion	endpoint	Software version of the agent/client. Typically of the form major_v.minor_v.build_no (for Cisco Secure Client Network Visibility Module 4.9 or later). The logged in username on the physical device, in the form Authority\Principal, on the endpoint from which the network flow is generated. A domain user is in the same form as a user locally logged in to a device (for example, MYDOMAIN\aduser or SOMEMACHINE\localuseracct).
aliul	AdditionalLoggedInUsersList	flow	(Windows only) A list of logged in users on the device other than nvzFlowLoggedInUser, each in the form SessionType:AccountType:Authority\Principal For example, rdp:8001:ACME\JSmith console:0002:<machine>\Administrator **Note** This field is empty for non-system processes.
ampguid	AMPGUID	endpoint	Unique endpoint ID of Secure Endpoint (AMP)
cppuid	CreationParentProcessUniqueIdentifier	process	PUID of the Parent process which has created the said process
ctm	timestamp		Absolute timestamp of the endpoint data or interface record in milliseconds
da	DestinationIPv4Addres	flow	IPv4 address of the destination to where the flow was generated from the endpoint.
da6	DestinationIPv6Address	flow	IPv6 address of the destination to where the flow was generated from the endpoint.
dh	FlowDestinationHostName	flow	The Destination Domain of the Destination Address to where the network flow was sent to the endpoint.
dp	DestinationTransportPort	flow	Destination Port number to where the flow was generated from the endpoint.
ds	FlowDNSSuffix	flow	The DNS suffix configured on the network to which the user was connected when the network flow was generated on the endpoint.
eptag	EndpointTag	endpoint	User-defined string to differentiate between endpoint subgroups. Exported as part of the Endpoint Identity data record when handling Network Visibility Module Tagging feature.
eveuuid	EveUUID	EVE data, flow	16 byte Unique ID that ties EVE records with flow records
evefdc	EveFDC	EVE data	EVE blob returned by mercury library by fingerprinting client hello sent during TLS session

Field Key	Field Name	Field Context	Field Description
fd	FlowDirection	flow	The direction of the Flow observed at the endpoint. The two values defined are 0: ingress flow and 1: egress flow.
fems	FlowEndMsec	flow	Timestamp (in msec) of when the network flow was complete by the endpoint (for Cisco Secure Client NVM 4.9 or later).
fes	EndSeconds	flow	Timestamp of when the network flow was completed by the endpoint.
fsg	FlowReportStage	flow	Stage of the flow record. 0: End flow record, 1: Start flow record, or 2: Periodic/Intermediate flow record.
fsms	FlowStartMsec	flow	Timestamp (in msec) of when the network flow was initiated by the endpoint (for Cisco Secure Client NVM 4.9 or later).
fss	StartSeconds	flow	Timestamp of when the network flow was initiated by the endpoint.
fv	FlowVersion	flow	Network Visibility Flow (nvzFlow) version sent by the client.
hh	HTTPHost	flow	Contents of HTTP Host header for HTTP/1.1 traffic
hm	HTTPMethod	flow	Contents of HTTP Method for HTTP/1.1 traffic.
ht	HTTPTarget	flow	Contents of HTTP URL Target for HTTP/1.1 traffic.
ibc	InBytesCount	flow	The total number of bytes downloaded during a given flow on the endpoint at layer 4, not including L4 headers.
ii	InterfaceIndex	interface	The index of the network interface as reported by the operating system.
iid	InterfaceInfoUID	flow interface	Unique ID for an interface metadata. Use to look up the interface metadata from the InterfaceInfo records.
im	InterfaceMacAddress	interface	MAC address of the interface.
in	InterfaceName	interface	Network Interface/Adapter name as reported by the operating system.
ist	TrustState	interface	Parsed from InterfaceDetailsList, this is the STATE portion, for example STATE=Trusted. Either Cisco Secure Client VPN is active or has determned the device is on a Trusted Network based on TND.
it	InterfaceType	interface	Interface type, such as Wired, Wireless, Cellular, VPN, Tunneled, Bluetooth, and so on. Enumeration of network types.

Field Key	Field Name	Field Context	Field Description
liuid	LoggedInUser	flow	If this field is empty, it indicates that either no user was logged into the device, or a user remotely logged into the device via RDP, SSH, and so on. liuida and liuidp fields would also be empty as a result. User information can still be derived from the process account information.
liuida	LoggedInUserAuthority	flow	The authority portion of the logged in username on the physical device from which the network flow is generated.
liuidp	LoggedInUserPrincipal	flow	The principal portion of the logged in username on the physical device from which the network flow is generated.
luat	LoggedInUserAccountType	flow	Account type of the logged in user, as per the enumeration of AccountType.
mhl	ModuleHashList	flow	List of 0 or more SHA256 hashes of the modules associated with the nvzFlowModuleNameList.
mnl	ModuleNameList	flow	List of 0 or more names of the modules hosted by the process that generated the flow. This name can include the main DLLs in common containers such as dllhost, svchost, rundll32, and so on. It can also contain the other hosted components such as the name of the jar file in a JVM.
obc	OutBytesCount	flow	The total number of bytes uploaded during a given flow on the endpoint at layer 4, not including L4 headers.
osn	OSName	endpoint	Name of the operating system on the endpoint (for example, WinNT). This name matches the value sent by AnyConnect VPN to an ASA.
osv	OSVersion	endpoint	Version of the operating system on the endpoint (for example, 6.1.7601). This version matches the value sent by AnyConnect VPN to an ASA.
pa	ProcessAccount	flow	The fully qualified account, in the form Authority\Principal, under whose context the application generating the Network Flow on the endpoint was executed.
paa	ProcessAccountAuthority	flow	The authority portion of the fully qualified account under whose context the application generating the Network Flow on the endpoint was executed.
pap	ProcessAccountPrincipal	flow	The principal portion of the fully qualified account under whose context the application generating the Network Flow on the endpoint was executed.
parg	ProcessArgs	flow	Command line arguments of the process that initiated the network flow (for Cisco Secure Client NVM 4.9 or later).
pct	ProcessCreationTimeStamp	process	Creation time of the process since Epoch in milliseconds.

Field Key	Field Name	Field Context	Field Description
ph	ProcessHash	flow	Unique SHA256 hash for the executable generating the Network Flow on the endpoint.
pid	ProcessId	flow	Process Id of the process that initiated the network flow (for Cisco Secure Client NVM 4.9 or later).
pil	ProcessIntegrityLevel	flow	Integrity level defines the trust between process and another object (files, processes, or threads).
pn	ProcessName	flow	Name of the executable generating the Network Flow on the endpoint.
ppa	ParentProcessAccount	flow	The fully qualified account, in the form Authority\Principal, under whose context the parent process of the application generating the Network Flow on the endpoint was executed.
ppaa	ParentProcessAccountAuthority	flow	The authority portion of the fully qualified account under whose context the parent process of the application generating the Network Flow on the endpoint was executed.
ppap	ParentProcessAccountPrincipal	flow	The principal portion of the fully qualified account under whose context the parent process of the application generating the Network Flow on the endpoint was executed.
pparg	ParentProcessArgs	flow	Command line arguments of the parent of the process that initiated the network flow (for Cisco Secure Client NVM 4.9 or later).
ppath	ProcessPath	flow	File system path of the process that initiated the network flow (for Cisco Secure Client NVM 4.9 or later).
pph	ParentProcessHash	flow	Unique SHA256 hash for the executable of the parent process of the application generating the Network Flow on the endpoint.
ppid	ParentProcessId	flow	Process Id of the parent of the process that initiated the network flow (for Cisco Secure Client NVM 4.9 or later).
ppil	ParentProcessIntegrityLevel	flow	Integrity level defines the trust between parent process and another object (files, processes, or threads).
ppn	ParentProcessName	flow	Name of the parent process of the application generating the Network Flow on the endpoint.
pppath	ParentProcessPath	flow	File system path of the parent of the process that initiated the network flow.
ppuat	ParentProcessAccountType	flow	Account type of the parent-process account, as per the enumeration of AccountType.
ppuid	ParentProcessUniqueIdentifier	process	PUID of the Parent Process which is reported by respective OS

Field Key	Field Name	Field Context	Field Description
pr	ProtocolIdentifier	flow	Network Protocol number associated with each flow. Currently, we only support TCP (6) and UDP (17).
puat	ProcessAccountType	flow	Account type of the process account, as per the enumeration of AccountType.
puid	ProcessUniqueIdentifier	flow process	Unique Identifier for the process.
qid	QueryID	Osquery data	8 byte unique ID for each query. First 4 bytes are reserved for the Enterprise Number; and the last 4 bytes are for query index.
qjr	QueryJsonResponse	OSquery Data	Paginated JSON response output.
qpi	CurrentPage	OSquery Data	Since JSON response for a query can be too large to send, Network Visibility Module paginates the JSON output. This field indicates which page is present in the current record.
qpn	TotalPages	OSquery Data	Total number of pages into which the JSON response is divided.
qt	QueryTimestamp	OSquerydata	Timestamp to correlate all paged records of osquerydata for a given UDID and qid
qv	OSqueryVersion	OSquery Data	Version of OSquery software running on the endpoint. Typically of the form major_v.minor_v.build_no
sa	SourceIPv4Address	flow	IPv4 address of the interface from where the flow was generated on the endpoint.
sa6	SourceIPv6Address	flow	IPv6 address of the interface from where the flow was generate on the endpoint
sm	SystemManufacturer	endpoint	Endpoint manufacturer (for example, Lenovo, Apple, and so on).
sp	SourceTransportPort	flow	Source port number from where the flow was generated on the endpoint.
SSID	SSID	interface	Parsed from InterfaceDetailsList, this is the SSID portion, for example SSID=internet.
st	SystemType	endpoint	Endpoint type (such as x86, x64).
udid	UDID	flow endpoint interface OSquerydata	A unique identifier for each endpoint in the network. It is derived and recreated from hardware attributes and is used to correlate records back to a single source. This matches the same value sent by AnyConnect VPN to an ASA.

Field Key	Field Name	Field Context	Field Description
vsn	VirtualStationName	endpoint	Device name configured on the endpoint (such as Boris-Macbook). Domain joined machines will be in the form [machinename].[domainname].[com](such as CESA-WIN10-1mydomain.com).

Host Firewall Recommendation for NVM Collector

The recommended Host Firewall settings on a Network Visibility Module Collector node are as follows:

- Inbound—Allow all Network Visibility Module Hosts' UDP traffic on the *netflow_collector_port* configured in the acnvm.conf file
- Outbound— Allow TCP traffic only to Splunk HEC collector IP(s) range and port, specified by the URL in acnvm.conf file.

  Note In continuation to outbound pointers.

- Outbound—Allow UDP traffic only to the configured *syslog_server_ip* (on *syslog_flowdata_server_port, syslog_sysdata_server_port, syslog_intdata_server_port, and syslog_processinfodata_server_port)* ports configured in the acnvm.conf file.
- Outbound—Allow TCP traffic to Splunk HEC collector IP(s) range and port specified by the URL in acnvm.conf.

All other incoming and outgoing traffic from the collector should be blocked unless it is needed by some other software on the Collector nodes.

Set Up Network Visibility Module Collector

To get the Network Visibility Module Collector set up, follow these steps:

1. Install or Upgrade Network Visibility Module Collector on Linux, on page 9.

   OR

   Build a Docker Image, on page 10.
2. Adjust Server Process of Hosts With Multiple Cores, on page 10.
3. Filter Network Visibility Module Collector Flows, on page 10.

Afterwards, you have the following Options for Collector Export, on page 11.

Install or Upgrade Network Visibility Module Collector on Linux

Procedure

Step 1 Download acnvmcollector-*version*.zip from Cisco Software Download Site.

Step 2 Extract the .zip file to any temporary directory.

Step 3 If this is a fresh install, change configuration settings according to Options for Collector Export, on page 11, Set Up Collector DTLS, on page 13, or Filter Network Visibility Module Collector Flows, on page 10. If this is an upgrade, existing configuration is retained.

Step 4 Execute the install.sh script with super user privileges.

Build a Docker Image

You can run the Network Visibility Module collector in a Docker Container. The acnvmcollector file includes a Dockerfile image. You should first make any configuration adjustments (as needed) to the acnvm.conf file before building a Docker image, as parameters for running the docker image depend on your acnvm.conf file. In the directory containing the Docker file, build the image:

```
docker build -t nvmcollector
```

Enter the following for the default configuration, where the collector listens on port 2055 and the syslog server is on the same host:

```
docker run -t -p 2055:2055/udp --net="host" nvmcollector
```

Adjust Server Process of Hosts With Multiple Cores

You can adjust the multi-core behavior of the Network Visibility Module Collector, as well as include or exclude filtering capabilities. This adjustment is primarily used if you performed a Linux installation. By default, when running on a host with multiple cores, the Collector creates a separate server process per core. You can adjust the process to force just one or to force two server processes. You also have the option to disable multi-core processes.

To disable multiprocessing and force a single process:

```
{
            "multiprocess":
                {"enabled": false}
}
```

To force two server processes:

```
{
    "multiprocess":
        {
            "enabled": true,
            "numProcesses": 2
        }
}
```

Filter Network Visibility Module Collector Flows

The Collector supports three optional flow filtering modes: inclusion, exclusion, and hybrid, as defined in a separate JSON policy file. You should specify the policy file path when starting the Collector, and the Collector will look for the policy stored in this file by default: /opt/acnvm/conf/acnvmfilters.conf.

If no policy exists, filtering is disabled, and all flows are processed and exported. The three filtering modes operate as follows:

- **Include Only** —By default, a flow is dropped unless it matches an include rule.
- **Exclude Only**—By default, a flow is collected unless it matches an exclude rule.
- **Include + Exclude (hybrid)**— By default, a flow is dropped unless it matches an include rule AND it does not match an exclude rule.

Each rule is specified as a JSON dictionary, with each key and value pair specifying a match criteria on a flow field (whose name matches the key). Suffix wildcards are supported for string field types and is denoted by an asterisk.

Example to exclude all DNS flows

```
{
    "rules":
        {
            "exclude":
            [
                {"dp": 53, "pr": 17}
            ]
        }
}
```

Example to exclude flows to certain DNS servers

```
{
     "rules":
        {
            "exclude":
            [
                {"dp": 53, "pr": 17, "da": "1.2.*"}
                {"dp": 53, "pr": 17, "da": "8.8.8.8"}
            ]
        }
}
```

Example to collect flows only from Angry Birds (Android app), but ignore DNS flows

```
{
    "rules":
        {
            "include":
            [
                {"pname": "com.rovio.angrybirds")
            ],
            "exclude":
            [
                {"dp": 53, "pr": 17, "da": "1.2.*"},
                {"dp": 53, "PRP: 17, "da": "8.8.8.8"}
            ]
        }
}
```

Options for Collector Export

The collector exporter currently supports syslog, Kafka, HEC Export, or your own exporter (with a custom plugin). HEC exporter sends out records in batches when either the batch size is reached or when the batch timeout (in milliseconds) is reached, whichever happens first.

Sample Syslog Export Configuration

```
{
    "exporter": {
        "type": "syslog",
        "syslog_server": "localhost",
        "flow_port": 20519,
        "endpoint_port": 20520,
        "interface_port": 20521,
        "osquerydata_port": 20522,
        "processinfodata_port":20523,
        "mercurydata_port":20524
```

```
    }
}
```

Sample Minimal Splunk HEC Export Configuration

```
{
    "exporter": {
        "type": "hec",
        "hec_url": "https://splunk.company.com:8443/services/collector",
        "hec_token": "12345678-1234-1234-1234-123456789012"
    }
}
```

hec_url is a Splunk HEC endpoint URL.

hec_token is a Splunk HEC authentication token

Sample Splunk HEC Export Configuration (with optional fields)

```
{
    "exporter": {
        "type": "hec",
        "hec_url": "https://splunk.company.com:9643/services/collector",
        "hec_token": "12345678-1234-1234-1234-123456789012",
        "hec_root_ca": "/opt/cisco/nvm/certs/splunk-ca.pem",
        "hec_batch_size": 200,
        "hec_batch_timeout": 15000,
        "hec_sourcetype": "cisco:nvm",
        "hec_index": "cisco_nvm"
    }
}
```

hec_root_ca is Splunk HEC's Root CA certificate file path

hec_batch_size is number of records per batch

hec_batch_timeout is batch timeout in milliseconds

Sample Kafka Export Configuration

```
{
    "exporter": {
        "type": "kafka",
        "bootstrap_server": "localhost:9092",
        "flow_port": "flow",
        "endpoint_port": "endpoint",
        "interface_port": "interface",
        "osquerydata_port": "osquerydata",
        "processinfodata_port": "processinfodata",
        "mercurydata_port": "mercurydata"
    }
}
```

Sample Custom Plugin

You can extend the collector's export capability using native C++ code by building a shared library against the plugin API. To use your custom plugin, a special configuration is required in the main collector configuration:

```
{
    "exporter": {
        "type": "plugin"
    }
}
```

Set Up Collector DTLS

You can configure the Network Visibility Module (NVM) to send data securely to the Collector, over DTLS. In the Network Visibility Module Profile Editor, when the *Secure* checkbox is checked, the Network Visibility Module uses DTLS as the transport. The DTLS server (collector) certificate must be trusted by the endpoint for the DTLS connection to work. Untrusted certificates are silently rejected. The minimum supported versions are DTLS 1.2. The Collector only works in one mode: either secure or unsecure.

The following certificate requirements must also be met:

- Collector certificates/certificate chains must be trusted by the client. (No configuration exists on Cisco Secure Client.)
- The certificate must be in PEM format.
- Certificate key password is not supported. (Cisco Identity Services Engine (ISE) and its internal Certificate Authority requires one.)
- You can use any certificate on the Collector as long as Cisco Secure Client trusts it. (For example, internal Public Key Infrastructure (PKI) and other well known certificates are trusted.)
- Cisco Secure Client NVM Profile Collector configuration must be set to IP or FQDN, based on what the common name (CN) of the certificate uses. FQDN is always preferred in the case of IP address changes. If you use an IP address, the Collector certificate CN or Subject Alternative Name (SAN) must have that IP. If you have FQDN as the CN in the certificate, the Network Visibility Module profile must have the same FQDN as a Collector.

Restart the Cisco Secure Client Network Visibility Module service after the configuration file is updated. Any profiles pushed from ISE or ASA require a disconnect and reconnect to the network.

Configure Collector for DTLS

Follow these steps on the device that hosts the collector:

Before you begin

Read the Set Up Collector DTLS, on page 13 section.

Procedure

Step 1 Create a **/opt/acnvm/certs** directory.

Step 2 Save the certificate and key in the /opt/acnvm/certs directory so that you can apply the certificate to the collector. Ensure that the certificate and private key file are in pem format.

Step 3 Change the owner and group of the folder to **acnvm:acnvm** with the following command:**sudo chown -R acnvm:acnvm certs/:**

Step 4 Set the permission as 400 for the certificate and private key files under /opt/acnvm/certs with the following command: **sudo chmod 400 ***.

Step 5 Configure the acnvm.conf section with the certificate and key.

Step 6 After the configuration and certificate are in place, restart the collector:**sudo systemctl restart acnvm.service**

Step 7 Check the collector status with the **sudo systemctl status acnvm.service** command:

```
{
    "security": {
```

```
        "dtls_enabled": true,
         "server_certificate":"opt/acnvm/certs/public.pem",
         "server_pkey":"/opt/acnvm/certs/private.pem",
    }
```

The rest of the configuration looks as follows:

```
"syslog_server_ip" : "192.0.2.113",
        "syslog_flowdata_server_port" : 20519,
        "syslog_sysdata_server_port" : 20520,
        "syslog_intdata_server_port" : 20521,
        "netflow_collector_port" : 2055
    },
```

Step 8 Run the **install.sh** script with superuser privileges:**sudo ./install.sh**.
The account needs sudo permissions or root to run the install.sh script, as well as permissions for the acnvm service account.

Set Up Collector mDTLS

You can configure the Network Visibility Module and Network Visibility Module Collector to run in mDTLS mode. Besides the client verifying the server's identity (DTLS), the server can also verify a client's identity in mDTLS mode. This mode ensures secure data transmission before establishing a connection.

If the *Client Authentication* checkbox is enabled in the Network Visibility Module Profile Editor, and the certificate matching criteria is configured, the Network Visibility Module uses mDTLS as the transport. For the mDTLS connection, client certificates much be trusted by the server. Both the Collector and all clients operate in the same mode (unsecure, DTLS, or mDTLS).

Configure Collector for mDTLS

Follow the steps below to run in mDTLS mode.

Before you begin

Read Set Up Collector mDTLS, on page 14.

Procedure

Step 1 Save the root CA certificate of clients to the "/opt/acnvm/certs" directory.

Step 2 Change the owner and group of the directory from "opt/acnvm/certs/" to acnvm:acnvm with the following command:

sudo chown -R acnvm:acnvm certs/:

Step 3 Set the permission as 400 for the certificate and private key files under /opt/acnvm/certs with the following command:

sudo chmod 400 *.

Step 4 Update the acnvm.conf security section to add the following:

```
"security":
    {
        "dtls_enabled": true,
```

```
        "mtls_enabled": true,
        "ocsp_strict": false,
        "root_ca": "/opt/acnvm/certs/root_ca.pem",
        "server_certificate":"opt/acnvm/certs/public.pem",
        "server_pkey":"/opt/acnvm/certs/private.pem",
    }
```

For mDTLS, both dtls_enabled and mtls_enabled must be set to *true*. Server certificate and server private key should be present in files /opt/acnvm/certs/public.pem and /opt/acnvm/certs/private.pem respectively. Root CA for client certificates must be present in file /opt/acnvm/certs/root_ca.pem.

The Collector will also verify the OCSP status of client certificates. The ocsp_strict tag determines if the Collector should disallow connections from clients whose OCSP status cannot be verified. When ocsp_strict is set to false, the Collector runs in fail-open mode, allowing connections even if OCSP verification fails. When set to true, connections are only allowed if the OCSP status of the client certificate is successfully verified and is not revoked.

Step 5 After the configuration and certificates are in place, restart the Collector with the following command:

sudo systemctl restart acnvm.service

Step 6 Check the collector status with the **sudo systemctl status acnvm.service** command.

Step 7 Run the install.sh script with superuser privileges:

sudo ./install.sh.

The account must have sudo permissions or root access to run the install.sh script, as well as permissions for the acnvm service account.

Validate Installation of Network Visibility Module

After successful installation, Network Visibility Module should be listed in Installed Modules, within the Information section of Cisco Secure Client. Verify if the Network Visibility Module service is running on the endpoint, and if the profile is in the required directory.

Validate Collector Status

Ensure that the Collector is running, verifying that the collector is receiving IPFIX/cflow from the endpoints at all times. If it is not running, ensure that the acnvm account permissions for the file allow it to execute: /opt/acnvm/bin/acnvmcollector.

Collector Diagnostic Tool

The Collector Diagnostics Tool is a command line tool used to collect data for troubleshooting Network Visibility Module Collector installation and connection problems. It assembles the logs, status, and diagnostic information for Cisco Technical Assistance Center (TAC) analysis. You run the acnvmcolldiag tool on the device where the NVM Collector is running, and you launch it from the command line as a super user to collect diagnostic information.

Run the Collector Diagnostic Tool

The tasks performed by the command below are dependent on the configuration file present in the /opt/acnvm/conf/acnvmcolldiagconf.json file.

Procedure

Step 1 Enter **/opt/acnvm/bin/acnvmcolldiag - p** ***<path of directory to save diagnostic results>*** to launch the Collector Diagnostic Tool.

Step 2 Verify that a zip file with a *acnvmcolldiag* prefix was created and stored in the provided path.

Basic Troubleshooting

If results aren't as expected, check the following:

- Network connectivity between the client endpoint and the collector.
- Network Visibility Module installation on the client endpoint.
- If captures on the endpoint show IPFIX traffic being generated.
- If captures on the collector show IPFIX traffic being received and forwarded.
- If captures on third-party collector show it received traffic.
- For DTLS, Cisco Secure Client clients should trust the Collector certificate and have the Network Visibility Module profile enabled as secure. Also, the Collector must be configured for certificates. If you are running DTLS between the client and the Collector, you need to filter on DTLS traffic within Wireshark.

Network Visibility Module Database Size is Expanding

If you notice the Network Visibility Module database size is expanding under C:/%ProgramData%\Cisco\Cisco Secure Client, the logs aren't being sent from the client. The Network Visibility Module folder and the sql database show the size increase and further indicate that data is not being sent to the Collector. How Network Visibility Module caches, and the controls around caching, are described in the Network Visibility Module chapter of the Cisco Secure Client Administrator Guide.

Configuration Requirements for a Trusted Network

Cisco Secure Client Network Visibility Module sends flow information only when it is on a trusted network. It uses the Trusted Network Detection (TND) feature of the Cisco Secure Client to learn if the endpoint is on a trusted network or not. The network state is used by Network Visibility Module to determine when to export Network Visibility Module data and how to apply the appropriate Data Collection Policy. Network Visibility Module has its own TND configuration, which involves an SSL probe being sent to the configured trusted headend, expecting a certificate in response (if reachable). Network Visibility Module TND is configured in the NVM Profile Editor. If Network Visibility Module TND is not configured, it relies on the VPN module's TND configuration.



Note When operating from outside your internal network, TND makes DNS requests and attempts to establish an SSL connection to the configured server. Cisco strongly recommends the use of an alias to ensure that the name and internal structure of your organization are not revealed through these requests by a device outside of your internal network.

Trusted Network Detection is configured in the Cisco Secure Client Profile (xml) used for VPN, regardless of whether the VPN component is used in the environment or not. TND is enabled by configuring the Automatic VPN Policy section in the Cisco Secure Client Profile Editor, Preferences (Part 2). VPN's TND uses the information received via DHCP: domain name and DNS server. If

the DNS server and/or domain name match the configured values, the network is deemed as trusted. VPN also supports TLS certificate-based TND detection. You determine the action for Cisco Secure Client to take when the client is on a trusted network. For example, you can set the Trusted and Untrusted Network Policy to **Do Nothing**.

An incorrect TND configuration causes issues with Network Visibility Module. Perform these actions for expected Trusted Network Detection performance:

- Ensure the TND configuration is correct. Network Visibility Module exports only when on a trusted network. An example of an invalid TND configuration is having three DNS servers but not having three defined.
- Remove the trusted domain from the TND VPN configuration.
- Always include the Collector's IP address in the split include configuration for VPN. If the Collector's IP address is not part of the split tunnel and is untrusted, the data is sent out to the public interface.
- Ensure the CollectionMode is configured to collect on the current network (trusted or untrusted).
- Make sure that the VPN.xml and NVM_ServiceProfile.xml are in the correct folders and then restart.
- Start and then stop all Cisco Secure Client services.
- Bounce the network connected to the inside that has a connection to DNS server.
- TND detection behind a proxy is not supported.

Collection of Logs at the Client/Endpoint Side

To troubleshoot what Cisco Secure Client is doing, run the Diagnostic and Reporting Tool (DART) on the Network Visibility Module components. All logs needed for Network Visibility Module are handled by DART. It collects log files, configuration, and so on. Windows logs are in various places. Look in the event viewer for Network Visibility Module under Cisco Secure Client. macOS and Linux logs are found by filtering for nvmagent.

Network Visibility Module Collector Fails to Install

If you receive an *Acnvm.conf error:line number 17: expected key string* message in the system log directory while installing Collector or running the install script, check for an errant or extra comma.

Network Visibility Module Collector Fails to Start

If the code failed to execute on the acnvmcollector file /opt/acnvm/bin/acnvmcollector, the user and group might not have eXecute for the acnvmcollector.

Logging Levels and Collector Version

You can get the version of the collector with the **/opt/acnvm/bin/acnvmcollector -v** command.

To set the logging level to debug, use **log4cplus.rootLogger=DEBUG, STDOUT, NvmFileAppender** in the acnvmlog.conf file. The default and recommended logging level is *INFO*.

DTLS Issues

DTLS not configured— Indicates that it wasn't in the acnvm.conf.file.

Server key is invalid— Indicates the password key combination is not supported.

Related Documentation for Network Visibility Module Collector

Refer to the following documentation for additional resources:

- Cisco Secure Client Administrator Guide, Release 5.x, Network Visibility Module chapter —a more detailed description of the Network Visibility Module and its associated profile editor and collection parameters
- Cisco Network Visibility Solution Community Page —a Splunk guide for those using Cisco Endpoint Security Analytics (CESA)
- CESA Built On Splunk Quickstart POV Kit and Deployment Guide— how users of Cisco Endpoint Security Analytics (CESA) can set up a proof of value or production deployment
- Cisco Endpoint Security Analytics (CESA) Dashboard Overview and FAQ—what CESA users need to interpret the dashboard
- Splunk Documentation—how to set up and use HTTP Event Collector

© Cisco Systems, Inc. All rights reserved.

Americas Headquarters
Cisco Systems, Inc.
San Jose, CA 95134-1706
USA

Asia Pacific Headquarters
CiscoSystems(USA)Pte.Ltd.
Singapore

Europe Headquarters
CiscoSystemsInternationalBV
Amsterdam,TheNetherlands

Cisco has more than 200 offices worldwide. Addresses, phone numbers, and fax numbers are listed on the Cisco Website at www.cisco.com/go/offices.