

Streaming Telemetry

In addition to monitoring the device using the integrated dashboards and CLI, you can configure the device to send data to your telemetry collector. You can use the collector to monitor multiple devices in your network.

The following topics explain the requirements for using streaming telemetry and how to set up the telemetry collector and its connection to the Firewall Threat Defense device.

- About Streaming Telemetry, on page 1
- Guidelines for Streaming Telemetry, on page 1
- Enable Streaming Telemetry, on page 2
- Setting Up the Telemetry Collector, on page 6
- Troubleshooting Telemetry Streaming, on page 10

About Streaming Telemetry

You can configure the device to send system health and telemetry data to an external telemetry collector that uses Google Remote Procedure Calls (gRPC) to collect data. You can then use your telemetry collector to monitor the device and integrate with your custom telemetry solution.

The connection between the device and telemetry collector uses Mutual Transport Layer Security (mTLS) Authentication to ensure security. The device and telemetry collector exchange certificates to verify client and server identities, and they encrypt the data transmission. The device initiates connections to the telemetry server (the dial-out model).

Guidelines for Streaming Telemetry

Keep the following guidelines in mind when configuring streaming telemetry:

- You can use IPv4 addresses only.
- The certificates used for the Firewall Threat Defense device and the telemetry collector should be signed by the same Certificate Authority (CA) to ensure communication. Create the necessary certificate for the Firewall Threat Defense device (or reuse the one configured for the Firewall Threat Defense web server), and download the certificate used by the telemetry collector.
- Each Firewall Threat Defense device can connect to a single telemetry collector only. But a single collector can be used for multiple Firewall Threat Defense devices.

- For devices configured for high availability, you must configure streaming telemetry on each device separately. The telemetry configuration is not replicated from the active to the standby unit. If desired, you can configure the primary and secondary units to use different telemetry collectors.
- The Firewall Threat Defense device uses the following ports for streaming telemetry.
  - Control channel: 9276 over HTTP
  - Data channel: 8087 over HTTP
  - Ports 9273 and 9276 are used for diagnostics
- You can write remote procedures to configure the Firewall Threat Defense device from your telemetry collector using the following Firewall Threat Defense APIs:
  - /object/internalcertificates
  - /object/externalcacertificates
  - /object/networks
  - /devicesettings/default/telemetrystreamingconfig

Enable Streaming Telemetry

After you have set up a telemetry collector, you can configure a connection between the Firewall Threat Defense device and the telemetry server. Once the connection is configured, the Firewall Threat Defense device automatically tries to set up the connection and if successful, streams data as frequently as the collector requests.

Before you begin

Ensure that the telemetry collector meets the requirements described in Setting Up the Telemetry Collector, on page 6.

Procedure

Step 1 Select **Objects** > **Certificates**, then **+** > **Add Internal Certificate**, and upload the client certificate that will be used by the Firewall Threat Defense device for secure communication. See Uploading Internal and Internal CA Certificates for detailed information.

Step 2 Select **Objects** > **Certificates**, then **+** > **Add Trusted CA Certificate**, and upload the CA certificate that will be used by the Firewall Threat Defense device to verify the identity of the collector. See Uploading Trusted CA Certificates for detailed information.

Step 3 Select **Objects** > **Network**, then **+**, to create the network object that identifies the telemetry collector. See Configuring Network Objects and Groups for detailed information.

Either create a Host object with the IPv4 address of the collector, or an FQDN object that includes the fully qualified name of the collector, such as telemetry.domain.com. An FQDN must resolve to an IPv4 address, and you must also configure DNS so the name can be correctly translated.

Step 4 Obtain the ID of the network object.

a) Select **API Explorer** from the more options button ⋮ to access the API pages.
b) Under **NetworkObject**, select **GET /object/networks**.
c) In the **Filter** field in the **Parameters** section, filter the output to equal the object name. For example, if the network object you created is TelemetryCollector, the filter would be:

   name:TelemetryCollector

d) Scroll to the bottom of the GET /object/networks section and click **Try It Out**.
e) If the call is correct, you should get a 200 response code and a meaningful object body, such as the following. Look for the id entry and make note of the value. In this example, the id value is **79ee2ea9-76b7-11ef-9515-f5b34b7d9531**.

```
{
  "items": [
    {
      "version": "p4qjmqtqn5c5e",
      "name": "TelemetryCollector",
      "description": null,
      "subType": "HOST",
      "value": "10.1.1.1",
      "isSystemDefined": false,
      "dnsResolution": "IPV4_AND_IPV6",
      "id": "79ee2ea9-76b7-11ef-9515-f5b34b7d9531",
      "type": "networkobject",
      "links": {
        "self":
"https://ftd1.domain.com/api/fdm/v6/object/networks/79ee2ea9-76b7-11ef-9515-f5b34b7d9531"

      }
    }
```

Step 5 Obtain the ID of the internal certificate object.

a) In the API Explorer, under **Certificate**, select **GET /object/internalcertificates**.
b) In the **Filter** field, filter by the certificate name. For example, if the internal certificate for the Firewall Threat Defense device is FTD1Cert, the filter would be:

   name:FTD1Cert

c) Scroll to the bottom of the GET /object/internalcertificates section and click **Try It Out**.
d) If the call is correct, you should get a 200 response code and a meaningful object body, such as the following. Look for the id entry and make note of the value. In this example, the id value is **d874dfa3-7423-11ef-b3a0-09429aedc3d3**.

```
{
  "items": [
    {
      "version": "gr573izgdsj2o",
      "name": "FTD1Cert",
…
      ATTRIBUTES REMOVED
…
      "id": "d874dfa3-7423-11ef-b3a0-09429aedc3d3",
      "type": "internalcertificate",
      "links": {
        "self":
"https://ftd1.domain.com/api/fdm/v6/object/internalcertificates/d874dfa3-7423-11ef-b3a0-09429aedc3d3"
```

```
      }
    }
```

Step 6 Obtain the ID of the trusted CA certificate object.

a) In the API Explorer, under **Certificate**, select **GET /object/externalcacertificates** .

b) In the **Filter** field, filter by the certificate name. For example, if the trusted CA certificate for the telemetry collector is TelemetryCollectorCert, the filter would be:

name:TelemetryCollectorCert

c) Scroll to the bottom of the GET /object/externalcacertificates section and click **Try It Out**.

d) If the call is correct, you should get a 200 response code and a meaningful object body, such as the following. Look for the id entry and make note of the value. In this example, the id value is **c3d925b4-7423-11ef-b3a0-bf815c0136ac**.

```
{
  "items": [
    {
      "version": "fkry47nobvcnu",
      "name": "TelemetryCollectorCert",
…
      ATTRIBUTES REMOVED
…
      "id": "c3d925b4-7423-11ef-b3a0-bf815c0136ac",
      "type": "externalcacertificate",
      "links": {
        "self":
"https://ftd1.domain.com/api/fdm/v6/object/externalcacertificates/c3d925b4-7423-11ef-b3a0-bf815c0136ac"

      }
    }
```

Step 7 Configure the connection between the Firewall Threat Defense device and the telemetry collector.

a) In the API Explorer, under **TelemetryStreamingConfig**, select **POST /devicesettings/default/telemetrystreamingconfig.**

b) Under **Parameters** > **Body**, type in the following template into the **Value** edit box (to avoid copying hidden invalid characters). The meaning of each field is explained in this template. Explanations within <> characters are variables that you must replace; the other values need to remain as shown. Comma, parentheses, colon, and {} placement are critical.

```
{
"name": "<a unique name for the gRPC streaming config API>",
"connectionMode": "DIAL_OUT",
"port": "<port on which the collector is waiting for connections from the
          Threat Defense device, 1-65535. Check the collector configuration
          for the right value.>",
"targetHost": {
    "name": "<name of the network object that identifies the telemetry collector host>",

     "id": "<ID of the network object>",
     "type": "networkobject"
},
"clientCertificate": {
      "name": "<The name of the internal certificate that identifies the Threat Defense
 device>",
      "id": "<ID of the internal certificate object.>",
      "type": internalcertificate
}
```

```
"caCertificate": {
      "name": "<The name of the trusted CA certificate for the telemetry collector>",
       "id": "<ID of the trusted CA certificate>",
      "type": "externalcacertificate"
},
"type": "telemetrystreamingconfig"
}
```

Example:

Given the example values shown in this procedure, the following would be a correct payload. Note that the name and port values are not determined by the previous steps, you can change these as needed.

```
{
"name": "YourCompanyTelemetry",
"connectionMode": "DIAL_OUT",
"port": 50051,
"targetHost": {
   "name": "TelemetryCollector",
   "id": "79ee2ea9-76b7-11ef-9515-f5b34b7d9531",
   "type": "networkobject"
},
"clientCertificate": {
    "name": "FTD1Cert",
    "id": "d874dfa3-7423-11ef-b3a0-09429aedc3d3",
    "type": "internalcertificate"
},
"caCertificate": {
     "name": "TelemetryCollectorCert",
     "id": "c3d925b4-7423-11ef-b3a0-bf815c0136ac",
     "type": "externalcacertificate"
},
"type": "telemetrystreamingconfig"
}
```

c) Scroll to the bottom of the section and click **Try It Out**.

d) Look for a response code of 200. If you see any other code, fix the errors and try again. The successful response body should look like the following:

```
{
  "version": "jfwu476cue32n",
  "name": "YourCompanyTelemetry",
  "connectionMode": "DIAL_OUT",
  "port": 50051,
  "targetHost": {
    "version": "p4qjmqtqn5c5e",
    "name": "TelemetryCollector",
    "id": "79ee2ea9-76b7-11ef-9515-f5b34b7d9531",
    "type": "networkobject"
  },
  "clientCertificate": {
    "version": "gr573izgdsj2o",
    "name": "FTD1Cert",
    "id": "d874dfa3-7423-11ef-b3a0-09429aedc3d3",
    "type": "internalcertificate"
  },
  "caCertificate": {
    "version": "fkry47nobvcnu",
    "name": "TelemetryCollectorCert",
    "id": "c3d925b4-7423-11ef-b3a0-bf815c0136ac",
    "type": "externalcacertificate"
  },
```

```
  "id": "b6dc6f28-76c1-11ef-9515-8ff976794f92",
  "type": "telemetrystreamingconfig",
  "links": {
    "self":
"https://ftd1.domain.com/api/fdm/v6/devicesettings/default/telemetrystreamingconfig/b6dc6f28-76c1-11ef-9515-8ff976794f92"

  }
}
```


What to do next

See the following topics on how to verify that telemetry streaming is working correctly:

- Checking the Status of the Telemetry Streaming Service, on page 10
- Verifying the Telemetry Collector Receives Data, on page 12

Setting Up the Telemetry Collector

You must supply your own telemetry collector (either off-the-shelf or custom built) to receive telemetry data from the Firewall Threat Defense device, aggregate the information, and display it in a meaningful way to meet your organization's operational requirements. The following provides some general information about setting up a telemetry collector to use grip calls to collect data from the Telegraph component running on the Firewall Threat Defense device.

Procedure

Step 1 Ensure that your telemetry collector meets the requirements listed in Guidelines for the Telemetry Collector, on page 6.

Step 2 Configure the proto definition as explained in Proto Definition on the Telemetry Collector, on page 7.

Step 3 Ensure that the telemetry collector can receive and respond to the Telegraph client running on the Firewall Threat Defense device as explained in Communication between the Firewall Threat Defense Device and Telemetry Collector, on page 8.

Guidelines for the Telemetry Collector

- You can run the client on a Windows, Mac, Linux, or Unix server.
- You must install Go on the telemetry collector. Minimum Go version is 1.20.
- The telemetry collector must have an IPv4 address, and there must be proper routing, either directly or through a proxy, with the Firewall Threat Defense devices that use it. If connectivity is lost, the Firewall Threat Defense device retries the connection every 5 minutes.
- The listening port on the telemetry collector must be a valid TCP port (1-65535) that is not already reserved for other purposes.

- The server certificate, server key, and CA certificate on the telemetry collector must be located at the following paths:
  - Server Key: /root/grpc-certs/keys/server.key
  - Server Certificate: /root/grpc-certs/keys/server.crt
  - CA Certificate: /root/grpc-certs/keys/ca.crt
- When certificates expire, you will see authentication errors on the telemetry client, and streaming will cease. You need to replace the certificates to correct the authentication problem and resume transmission.
- Messages use the Prometheus time series format. Your client must be able to handle this format.
- The following telemetry collector is not supported: https://github.com/CiscoSE/grpc_collector.

Proto Definition on the Telemetry Collector

The Firewall Threat Defense device uses protocol buffers for structuring data. The proto definition on the telemetry collector should contain the following.

```
syntax = "proto3";
// Update the go_package option to a local package path
option go_package = "grpcstreaming/grpc_streaming_proto";
package proto;
service GrpcStreamingService {
    rpc DataStream (stream DataResponse) returns (stream DataRequest);
    rpc ControlStream (stream ControlResponse) returns (stream ControlRequest);
}
message ControlResponse {
    string version = 1;
    string ftd_uuid = 2;
    string hostname = 3;
    bool init_streaming = 4;
    repeated string capabilities = 5; // ['metric_streaming']
    // cancel stream acknowledgement
    bool ack = 6;
}
message ControlRequest {
    string version = 1;
    int64 interval = 2;
    repeated string metric_subscriptions = 3;
    // cancel stream
    StreamCancellationMessage cancellation_message = 4;
}
message DataResponse {
    string ftd_uuid = 1;
    repeated Metric metrics = 2;
}
message DataRequest {
    bool ack = 1;
}
message StreamCancellationMessage {
    string collector_uuid = 1;
    bool cancel_stream = 2;
}
message Tag {
    string key = 1;
    string value = 2;
}
```

```
message Metric {
    int64 timestamp = 1;
    string metricFamily = 2;
    double value = 3;
    repeated Tag tags = 4;
    string metricType = 5; //Counter | Gauge
}
```

Communication between the Firewall Threat Defense Device and Telemetry Collector

To initially establish the connection, the Firewall Threat Defense device sends the telemetry collector a gRPC request that includes a payload indicating initial_request=True. The request also advertises its capabilities, which is “metric_streaming.”

The telemetry collector needs to acknowledge the connection and respond with a message that includes a time interval at which collector is ready to receive streamed data. This interval signifies the expected frequency at which the Firewall Threat Defense device should stream telemetry to the collector, from 1 minute (60 seconds) to 24 hours. Once a valid frequency is obtained, the system sends an initial set of metrics, then provides additional information at the requested rate.

Following are the RPC methods used for communication:

- Unary RPC (Control Messages), sent from the Telegraf component on the Firewall Threat Defense device to configure streaming.

  rpc ConfigureMetricStreaming (TelegrafControlMessage) returns (CollectorControlMessage);

- Streaming RPC (Data Messages), which facilitate bi-directional streaming of metric data between Telegraf on the Firewall Threat Defense device and the telemetry collector.

  rpc BiDirectionalMetricStreaming (stream TelegrafDataMessage) returns (stream CollectorDataMessage);

The following topics go into more detail about the messages the telemetry collector should receive from the device, and the messages the collector must send to the device.

Telegraf Control Message (Control Channel)

The Telegraph control message is sent from the Telegraf component on the Firewall Threat Defense device to the collector to initiate metric streaming. It includes the init_request flag set to true.

```
message TelegrafControlMessage {
// Indicates the proto version used by the FTD device. First version will be 1.0
    string version;

// Indicates the device id of the sender
    string device_uuid;

// Indicates the device hostname of the sender
    string hostname;

//list of strings indicating the capabilities of FTD. This will be “metric_streaming”
    repeated string capabilities;

// Flag to initiate a collector response for configuring telemetry streaming
    bool init_streaming = 1;
}
```

Collector Control Message (Control Channel)

The collector control message is sent from the collector to the Firewall Threat Defense device as a response to the Telegraf control message. It includes the interval, which is the desired frequency of metric batches, from 1 minute (60 seconds) to 24 hours. The metric subscriptions component is optional.

```
message CollectorControlMessage {
// Indicates the proto version used by the target. Current version supported is 1.0
    string version;
// Time interval at which the FTD device should send metric batches
    int64 interval = 1;
// Set of metric families to subscribe to, the default value is the only supported value.
// Default: "all"
    repeated string metricSubscriptions = 2;
}
```

Stream Cancelation Message (Control Channel)

The Stream cancellation message is used to cancel an existing telemetry stream between the Firewall Threat Defense device and the telemetry collector. Either the Firewall Threat Defense device or the collector can issue this message on the control channel. The receiver of the cancellation request needs to reply with an ACK message. After the cancellation is complete, the Firewall Threat Defense device retries the collector every 5 minutes until the collector accepts the new streaming request. To permanently end streaming, simply delete the streaming configuration on the Firewall Threat Defense device.

```
message StreamCancellationMessage {
// Indicates the proto version used by the FTD device. First version will be 1.0
    string version;

// Indicates the device id of the sender
    string device_uuid;
// This flag indicates that the cancel request is true
    bool cancel_request;
}
```

Telegraf Data Message (Data Channel)

The Telegraf data message contains a batch of metrics sent from the Firewall Threat Defense device to the collector. It includes a repeated field called metrics containing individual Metric messages.

```
message TelegrafDataMessage {
    // Batch of metrics sent by Telegraf
    repeated Metric metrics = 1;
}
```

Metric Messages (Data Channel)

The Telemetry data are the metrics collected from various components of the system, such as interfaces, CPU, memory, disk usage, and so forth, that are transmitted from the Firewall Threat Defense device to the telemetry collector for monitoring and analysis. The format of this data is defined by the Metric message on the Proto Definition.

The following is an example of a metric message that contains telemetry data:

```
METRIC=timestamp:1718257445000
metricFamily:"cpu"  value:0.7  tags:{key:"cpu"  value:"CPU))"}  tags:{key:"description"
```

```
value:"cpu_utilisation"}  tags:{key:"process"  value:"lina"}  tags:{key:"rcpu"
value:"x86_cpu0"} tags:{key:"uuid"  value:"7eb19498-2519-11ef-a8dd-b74b4d43a7e7"}
metricType:"Gauge"
```

Metric messages include the following fields:

- Timestamp (int64 timestamp)—The exact time at which the metric was recorded, expressed in epoch time.
- Metric Family (string metricFamily)—The system component or resource being measured, such as "cpu", "memory", "disk", "interface".
- Value (double value)—The numerical value of the metric. The interpretation of this value depends on the metric type. For example, CPU utilization percentage.
- Tags (repeated Tag tags)—Additional context about the metric. Each tag is a key-value pair, where the key is a descriptive label (e.g. "cpu", "process", "interface") and the value provides specific details (e.g. "CPU0", "lina", "GigabitEthernet0/0").
- Metric Type (string metricType)—The nature of the metric. It can be a "Counter", which accumulates over time (e.g., total packets sent), or a "Gauge", which represents a value at a specific point in time (e.g., CPU utilization).

Collector Data Message (Data Channel)

The collector data message must be sent from the collector back to the Firewall Threat Defense device to acknowledge the receipt of TelegrafDataMessage. It includes an ACK flag set to true for successful reception.

```
message CollectorDataMessage {
    // ACK for every metric batch
    bool ack = 1;
}
```

Troubleshooting Telemetry Streaming

The following topics explain how to troubleshoot telemetry streaming.

Checking the Status of the Telemetry Streaming Service

When you enable telemetry streaming, the configuration is pushed to the device immediately. The service restarts and connects to the telemetry server, assuming all values are correct and there is a path to the collector.

Procedure

Step 1 In the API Explorer, under **TelemetryStreamingConfig**, select **GET /operational/telemetrystreamingstatuses**.

Step 2 Click **Try It Out**.

Step 3 If the response code is 200, check the **state** value in the response body.

The ideal response is a CONNECTED state with no error messages.

If the state is DISCONNECTED, look at the error messages to determine the possible problems.

For example, the following error indicates that a connection could not be made. This error example was generated by providing an IP address that was not for a telemetry connector, hence the i/o timeout. This could also indicate that you provided the wrong port value. Note that errors are categorized into types and include a count of the number of times the error has occurred since the service was started.


```
"items": [
    {
      "state": "DISCONNECTED",
      "errors": [
        {
          "errorType": "StreamingErrors",
          "errorMessage": "2024-09-30 19:20:33.575156378 +0000 UTC m=+308552.647839001 :
rpc error: code = Unavailable desc = connection error: desc = \"transport: Error while
dialing: dial tcp 10.1.1.1:50051: i/o timeout\"",
          "errorCount": 3857,
          "type": "telemetryerror"
        }
```


Another common error besides i/o timeout would be “no route to host,” where you have a routing issue in the network or in the Firewall Threat Defense device configuration.

For more information on error types, see Status Error Categories, on page 11.

Status Error Categories

If there is an error in the telemetry streaming service, whether the service status is Connected or Disconnected, the telemetry streaming status information includes error messages related to the issues encountered. Error messages include a time stamp when the error occurred, an error code, and a description.

Messages are organized into the following categories.

Abort Error (Code 0)

Errors that result in the telemetry service aborting. If these errors persist, contact Cisco Technical Support.

System Error (Codes 1-10)

These errors typically mean that no host name or UUID was configured. This means telemetry streaming was not configured successfully. Please redo the configuration.

Buffer Error (Codes 11-20)

These errors relate to the metric buffers, such as metric conversion problems. If these errors persist, contact Cisco Technical Support.

Auth Error (Codes 21-30)

These errors indicate that there is a problem with the certificates. Evaluate the messages and resolve the certificate issues.

For example, if a certificate has expired, you need to upload a new valid certificate and re-enable the service. Make sure that all certificates are signed by the same Certificate Authority.

Streaming Error (Codes 31-40)

These errors pertain to the connection between the Firewall Threat Defense device and the telemetry collector. Problems might include the wrong IP address and port numbers, or DNS resolution problems. They can also relate to routing issues.

Fixes might require re-doing the telemetry streaming configuration or resolving a routing problem in the network. Routing/DNS problems might also indicate a transient problem, for example, if your upstream link or DNS server goes down.

Invalid Argument Error (Codes 41-50)

These errors relate to bad data returned in a message. For example, the wrong protocol version or an interval frequency that is out of bounds. These errors require fixes to the telemetry collector rather than the Firewall Threat Defense device. Note that the service status might still be Connected even if these errors appear. For example, the following error occurs because the interval frequency is out of bounds, 30 seconds whereas the minimum allowed is 1 minute.

```
"state": "CONNECTED",
      "errors": [
        {
          "errorType": "InvalidArgumentErrors",
          "errorMessage": "2024-09-30 19:20:33.575156378 +0000 UTC m=+308552.647839001
 : Possible proto version mismatch or invalid streaming interval - client=192.168.97.90,
 proto version=0.0.1, streaming interval=30s",
          "type": "telemetryerror"
        }
],
```

Verifying the Telemetry Collector Receives Data

Verify the telemetry collector is receiving information from the Firewall Threat Defense device. You should see relevant messages and data on the telemetry collector console. For example:

```
2024/08/19 11:08:58 D! [grpc_client] Streaming interval set to=1m0s
2024/08/19 11:08:58 D! [grpc_client] Starting listener, attempting to listen at address=:50051
 over tcp
2024/08/19 11:08:58 D! [grpc_client] CERT_PATH: /root/grpc-certs/keys
2024/08/19 11:08:58 D! FTD signalling listener running on port=8087
2024/08/19 11:08:58 D! [grpc_client] Collector server started at port=50051
2024/08/19 11:09:24 D! [grpc_client] DataStream RPC invoked
2024/08/19 11:09:24 D! [grpc_client] ControlStream RPC invoked
2024/08/19 11:09:24 D! [grpc_client] Receiving metrics from
device=firepower-7eb19498-2519-11ef-a8dd-b74b4d43a7e7
2024/08/19 11:09:24 D! [grpc_client] ControlStream - received done signal
2024/08/19 11:09:24 D! [grpc_client] RPC
context=&{device:firepower-7eb19498-2519-11ef-a8dd-b74b4d43a7e7 controlStream:0xc000024120
 dataStream:0xc000096020}
2024/08/19 11:14:24 D! [grpc_client] info - received metric batch of count=479 from
device=firepower-7eb19498-2519-11ef-a8dd-b74b4d43a7e7
2024/08/19 11:14:24 D! [grpc_client] METRIC=timestamp:1718257445000  metricFamily:"interface"
  value:35386  tags:{key:"duplex_mode"  value:"FULL"}  tags:{key:"interface"
value:"GigabitEthernet0/0"}  tags:{key:"interface_description"}  tags:{key:"interface_name"
  value:"inside_interface"}  tags:{key:"interface_type"  value:"GigabitEthernet"}
tags:{key:"mac_address"  value:"0050.5683.0a21"}  tags:{key:"uuid"
value:"7eb19498-2519-11ef-a8dd-b74b4d43a7e7"}  tags:{key:"description"  value:"input_packets"}
  metricType:"Counter"
```