APPENDIX C

FlowCollector Configuration and Control Protocol

The FlowCollector configuration and control protocol is a text-based (ASCII) messaging protocol that facilitates the remote configuration and control of the FlowCollector application. The protocol includes specifications for creating unsolicited event notifications (UENs), or trap messages. Use this appendix as a reference when creating applications that you want the FlowCollector to communicate with, or as a reference for use with the Cisco Network Data Analyzer application.

Note In order for an application to receive UENs from FlowCollector, the **nf.resources** file must be configured to specify the IP address and port (or the DNS system name and port) that will receive UENs. See the "Modifying FlowCollector Resources" section on page 5-30 for details on this option.

Overview

This appendix describes the FlowCollector configuration and control protocol, a message-based application layer protocol that allows for messaging from, and remote manageability of, the FlowCollector application. Messages exchanged between FlowCollector and remote applications fall under two broad categories:

- request/response pairs
- unsolicited event notifications (UENs) or trap messages.

A list of relevant terminology is covered first, followed by the message format. Next, a description of the mode of communication between the FlowCollector and remote applications is provided.

Terminology

For the purpose of this document, the NetFlow FlowCollector 3.*x* (referred to as FlowCollector hereafter) application is the server, and applications making control or configuration requests are clients. Currently, the primary client is the Cisco Network Data Analyzer (referred to as the Data Analyzer hereafter) application, which provides a central point to configure and control multiple, remote FlowCollector stations. The following two terms are used throughout this appendix:

1. An *event* represents an operational or environmental condition of interest. Examples of events are:
   a. Collector process has terminated.
   b. Disk space is running low.
2. A UEN (or trap) is an unsolicited message that reports the occurrence of an event on FlowCollector, and is sent to one or more recipients. Examples of a recipient include the Network Data Analyzer application and any custom applications that have been designed to communicate with FlowCollector.

Message Format

This section describes a simple string-based message format to facilitate communication between clients and FlowCollector. The protocol is string-based to avoid the inherent big-endian and little-endian problems associated with communication between machines of differing architectures.

The messages fall into the following three categories:

1. Request—a request is initiated by the client (Data Analyzer) and contains authentication information.
2. Response—FlowCollector services the request and responds with a response message.
3. Event (UEN)—an event notification is sent by FlowCollector to a list of prespecified recipients.

Each message consists of a <*seqnum*>, <*msgtype*>, <*[login:password:]checksum*> (in this order) and one or more character strings. The general syntax is:

<*seqnum*><*ws*><*msgtype*><*ws*><*[login:password:]checksum*><*endline*>

<*command*|*event*>[:<*argument*>] [\n<*char string*>]

where:

<*ws*> represents one or more white spaces, and <*endline*> represents end of line. The <*seqnum*> value is a randomly assigned positive integer to track messages. The third field provides authentication as well as detection if the message was altered. It consists of the identification of the FlowCollector user and its password (on the FlowCollector machine) and the bytecount (total number of bytes) of the message. Event messages do not contain the login:password portion. In a response message, FlowCollector includes checksum bytes to represent the number of bytes in the response.

The <*msgtype*> character string is one of the following:

- GET
- SET
- EVENT
- GET_RESPONSE
- SET_RESPONSE

- AUTH_ERROR
- SYNTAX_ERROR
- DOWN_ERROR
- GET_ERROR
- SET_ERROR

GET and SET message types indicate a request message. A request may contain more than one operation (such as modify multiple fields of a thread). EVENT indicates an event message. In the case of response messages, the <*seqnum*> value is the same as in the corresponding request (so that clients can handle multiple request/response pairs asynchronously).

GET_RESPONSE, SET_RESPONSE, AUTH_ERROR, DOWN_ERROR, GET_ERROR, and SET_ERROR indicate a response message. GET_RESPONSE and SET_RESPONSE messages may include optional lines starting with the keyword "**warn**:" or "**info**:". However, receiving a GET_RESPONSE or SET_RESPONSE means the request was successfully executed.

Each GET/SET request contains an authentication string that is used to validate the requester's identity. Authentication failure results in an AUTH_ERROR response. FlowCollector sends the same AUTH_ERROR response if the checksum does not match up with the total number of bytes received. If FlowCollector cannot process a request because of unavailability of its processing engines (that is, backend processes are not running), it sends a DOWN_ERROR response. If an error is detected in parsing a request, a corresponding SYNTAX_ERROR response is returned.

If FlowCollector cannot process a request because of operational reasons (for any reason other than AUTH_ERROR, DOWN_ERROR, or SYNTAX_ERROR), it returns a GET_ERROR or SET_ERROR response. Such an error response includes at least one line starting with the keyword "**error**:" followed by a description. A GET_ERROR response occurs when a request to fetch attributes of a nonexistent thread is received.

The second line of the message contains a command (event in the case of UENs), optionally followed by an argument. In request and response messages, the response copies this second line from the corresponding request. See Table C-1 for command and argument examples.

Table C-1 Body of a GET Request

Command	Argument
filter_definition	-
thread_definition	-
protocol_definition	-
aggregation_list	-
application_stats	-
export_source_list	-
thread_list	-
filter_list	-
protocol_list	-
known_srcports_list	-
known_dstports_list	-
known_srcasns_list	-
known_dstasns_list	-

Table C-1 Body of a GET Request

Command	Argument
filter_attributes	*filterid*
thread_attributes	*threadid*
protocol_attributes	*protocolid*

The commands **filter_attributes**, **thread_attributes**, and **protocol_attributes** retrieve descriptions of the requested entity (filter/thread/protocol). As shown in Table C-1, these three commands require an additional argument, which is the ID of the entity to be obtained. The body of the corresponding response message is identical to the representation of the corresponding entity in the configuration files of FlowCollector (you should refer to the syntax of these entities for keywords, legitimate values of a field, and so forth).

Use the following examples to clarify GET request usage:

- To retrieve attributes of a filter:

```
300 GET login:passwd:checksum
filter_attributes:ALLOW-WWW
```

The expected sample response (assuming ALLOW-WWW exists) is:

```
300 GET_RESPONSE login:passwd:newchecksum
filter_attributes:ALLOW-WWW
Permit Srcport  80
Permit Dstport 8
```

- To retrieve attributes of a thread:

```
301 GET login:passwd:checksum
thread_attributes:HMAT
```

The expected sample response (assuming HMAT exists) is:

```
301 GET_RESPONSE login:passwd:newchecksum
thread_attributes:HMAT
Aggregation HostMatrix
Period 20
Port 9991
State Active
DataSetPath /opt/CSCOnfc/Data
DiskSpaceLimit 0
FileRetain 0
```

- To retrieve attributes of a protocol:

```
303 GET login:passwd:checksum
protocol_attributes:WWW
```

The expected sample response (assuming WWW exists) is:

```
303 GET_RESPONSE login:passwd:newchecksum
protocol_attributes:WWW
Dstport 80 OR Srcport 80
Prot 6
```

Note In the second example, the **DiskSpaceLimit** and **FileRetain** attributes may be dropped or renamed because of the **MaxUsage** feature in FlowCollector 3.*x*. In addition, some new attributes may be added to the thread's definition, so this example should not be viewed as a definitive list of attributes fetched by this request.

Table C-2 lists all valid SET commands. Requests for **add_filter**, **add_thread**, **thread_attributes**, and **add_protocol** append the definition (a description of the attributes) in the same manner as the corresponding GET_RESPONSE messages do to convey a definition of filter, thread, or protocol. FlowCollector may respond to a SET request with a SET_RESPONSE or a SET_ERROR response. A SET_ERROR response occurs when the requested entity does not exist, for example.

Table C-2 Body of a SET Request

Command	Request
start_collector	*duration*
stop_collector	-
dump_config	-
add_filter	*filterid*
drop_filter	*filterid*
add_thread	*threadid*
drop_thread	*threadid*
modify_thread	*threadid*
add_protocol	*protocolid*
drop_porotocol	*protocolid*

Use the following examples to clarify SET request usage:

- To add a filter:

```
400 SET login:passwd:checksum
add_filter:DENY-TELNET
Deny Srcport  23
Deny Dstport 23
Permit  Dstaddr         10.0.0.0         255.255.255.255
```

The expected sample response (assuming DENY-TELNET does not exist) is:

```
400 SET_RESPONSE login:passwd:newchecksum
add_filter:DENY-TELNET
```

- To drop a thread:

```
401 SET login:passwd:checksum
drop_thread:CALLRECORD
```

The expected sample response (assuming CALLRECORD does not exist) is:

```
401 SET_ERROR login:passwd:newchecksum
drop_thread:CALLRECORD
error:thread ID not found
```

Event conditions and arguments are described in Table C-3.

Table C-3 Body of an Event UEN Message

Condition	Argument
collector_started	*timestamp*
collector_stopped	*timestamp*
disk_consumption	*partition name*
rcvd_unsolicited_packet	*ip addr*
max_active_threads_limit	-
incompatible_pkt_and_aggregation	-
datafile_not_written	*filename*

Communication

FlowCollector expects the request messages to come over a TCP socket. FlowCollector opens a TCP socket on a well-known port (user-configurable) and awaits messages from all clients on this port. It uses a separate UDP socket to send event messages. It maintains a static list of recipients (IP addresses) of events and sends an event message to all of them. The administrator of FlowCollector is responsible for maintaining this list.

FlowCollector sends UENs autonomously. FlowCollector does not expect a response, so reception of the message cannot be assumed. FlowCollector uses a UDP socket to send UENs to a prespecified list of recipients.

On receiving a UEN, a manager (a software module designed to receive UENs, or traps) is free to decide how to process it. Typically, it creates a new entry in the event log. Also, it may signal or distribute the received UEN to other modules. These modules may want to query the agent (FlowCollector) for additional information, or may try to correlate the received UEN with other UENs and status information.

Message List

This section describes all valid configuration and control messages. Messages are organized by requests and their corresponding responses (including errors). Some errors apply to all request messages. These error conditions indicate the following:

- Authentication failure (returns an AUTH_ERROR message)
- Collector processes unavailable (returns a DOWN_ERROR message)
- Syntax-related (returns a GET_ERROR or SET_ERROR message with “error:syntax error” string appended to the end).

GET and GET_RESPONSE and GET_ERROR

Filter Definition

Filter Definition ("name, value [, mask]") pairs of all possible attributes of a filter. Any restriction (for example, that a character string should not exceed 20 characters) or the possible values in case of an enum, are included in parentheses immediately after.

```
200 GET login:passwd:checksum
filter_definition
```

An expected sample response is:

```
200 GET_RESPONSE login:passwd:newchecksum
filter_definition
Filter string(20)
enum(permit,deny), Srcaddr, ipaddr, ipaddr
enum(permit,deny), Dstaddr, ipaddr, ipaddr
enum(permit,deny), Nexthop, ipaddr, ipaddr
enum(permit,deny), Srcport, ushort
enum(permit,deny), Dstport, ushort
enum(permit,deny), Srcinterface, ushort
enum(permit,deny), Dstinterface, ushort
enum(permit,deny), Prot, ushort
enum(permit,deny), Protocol, string(20)
enum(permit,deny), TOS, ushort
enum(permit,deny), SrcAS, ushort
enum(permit,deny), DstAS, ushort
```

Thread Definition

Thread Definition ("name, value") pairs of all possible attributes of a thread. Any restrictions, (for example, that a character string should not exceed 20 characters) or the possible values in case of an enum, are included in parentheses immediately after.

```
200 GET login:passwd:checksum
thread_definition
```

An expected sample response is:

```
200 GET_RESPONSE login:passwd:newchecksum
thread_definition
Thread, string(20)
Aggregation, string
Filter, string(20)
Port, ushort
Period, ushort
DataSetPath, string
State, enum(Active, Inactive)
Compression No
Binary No
MaxUsage 500
```

Protocol Definition

Protocol Definition ("name, value") pairs of all possible attributes of a protocol. Any restrictions, (for example, that a character string should not exceed 20 characters) or the possible values in case of an enum, are included in parentheses immediately after.

```
200 GET login:passwd:checksum
protocol_definition
```

An expected sample response is:

```
200 GET_RESPONSE login:passwd:newchecksum
protocol_definition
Protocol, string(20)
Srcport, ushort
Dstport, ushort
Prot, ushort
Supported Aggregation Schemes
List of supported aggregation schemes ("name, bitmask" pair).
200 GET login:passwd:checksum
aggregation_list
```

An expected sample response is:

```
200 GET_RESPONSE login:passwd:newchecksum
aggregation_list
RawFlows, bitmask
SourceNode, bitmask
DestNode, bitmask
HostMatrix, bitmask
SourcePort, bitmask
DestPort, bitmask
Protocol, bitmask
DetailDestNode, bitmask
DetailHostMatrix, bitmask
DetailInterface, bitmask
CallRecord, bitmask
ASMatrix, bitmask
NetMatrix, bitmask
DetailSourceNode, bitmask
DetailASMatrix, bitmask
RouterAS, bitmask
RouterProtoPort, bitmask
RouterSrcPrefix, bitmask
RouterDstPrefix, bitmask
RouterPrefix, bitmask
ASHostMatrix, bitmask
HostMatrixInterface, bitmask
DetailCallRecord, bitmask
```

Application Statistics

```
200 GET login:passwd:checksum
application_stats
```

An expected sample response is:

```
200 GET_RESPONSE login:passwd:newchecksum
application_stats
<to be filled>
```

List Export Sources

Lists sources from which FlowCollector has received one or more packets.

```
201 GET login:passwd:checksum
export_source_list
```

An expected sample response is:

```
201 GET_RESPONSE login:passwd:newchecksum
export_source_list
171.69.2.77
blab-gw
bldg-a
```

Possible error responses can be:

```
201 AUTH_ERROR login:passwd:newchecksum
export_source_list
```

Or:

```
201 DOWN_ERROR login:passwd:newchecksum
export_source_list
```

List Filters

```
202 GET login:passwd:checksum
filter_list
```

An expected sample response is:

```
202 GET_RESPONSE login:passwd:newchecksum
filter_list
allow_www
deny_datacenter_subnet
```

Possible error responses can be:

```
202 AUTH_ERROR login:passwd:newchecksum
filter_list

Or:
202 DOWN_ERROR login:passwd:newchecksum
filter_list
```

List Threads

```
202 GET login:passwd:checksum
thread_list
```

An expected sample response is:

```
202 GET_RESPONSE login:passwd:newchecksum
thread_list
HMATRIX
DETAILASM
```

Possible error responses can be:

```
202 AUTH_ERROR login:passwd:newchecksum
thread_list
```

Or:

```
202 DOWN_ERROR login:passwd:newchecksum
thread_list
```

List Protocols

```
203 GET login:passwd:checksum
protocol_list
```

An expected sample response is:

```
203 GET_RESPONSE login:passwd:newchecksum
protocol_list
TELNET
WWW
FTP-SERVER
FTP-CLIENT
SNMP
NNTP
```

Possible error responses can be:

```
203 AUTH_ERROR login:passwd:newchecksum
protocol_list
```

Or:

```
203 DOWN_ERROR login:passwd:newchecksum
protocol_list
```

List Known srcports

```
205 GET login:passwd:checksum
known_srcport_list
```

An expected response is:

```
205 GET_RESPONSE login:passwd:newchecksum
known_srcport_list
0, 1024
1025, 9999 :1K_9K_Port_Rng
10000, 19999 :10K_19K_Port_Rng
20000, 29999 :20K_29K_Port_Rng
30000, 39999 :30K_39K_Port_Rng
40000, 49999 :40K_49K_Port_Rng
50000, 59999 :50K_59K_Port_Rng
60000, 65535 :60K_65K_Port_Rng
```

Possible error responses can be:

```
205 AUTH_ERROR login:passwd:newchecksum
known_srcport_list
```

Or:

```
205 DOWN_ERROR login:passwd:newchecksum
known_srcport_list
```

List Known dstports

```
206 GET login:passwd:checksum
known_dstport_list
```

An expected response is:

```
206 GET_RESPONSE login:passwd:newchecksum
known_dstport_list
0, 1024
1025, 9999 :1K_9K_Port_Rng
10000, 19999 :10K_19K_Port_Rng
```

Possible error responses can be:

```
206 AUTH_ERROR login:passwd:newchecksum
known_dstport_list
```

Or:

```
206 DOWN_ERROR login:passwd:newchecksum
known_dstport_list
```

List Known srcasns

```
207 GET login:passwd:checksum
known_srcasns_list
```

An expected response is:

```
207 GET_RESPONSE login:passwd:newchecksum
known_srcasns_list
0, 16000
```

Possible error responses can be:

```
207 AUTH_ERROR login:passwd:newchecksum
known_srcasns_list
```

Or:

```
207 DOWN_ERROR login:passwd:newchecksum
known_srcasns_list
```

List Known dstasns

```
208 GET login:passwd:checksum
known_dstasns_list
```

An expected response is:

```
208 GET_RESPONSE login:passwd:newchecksum
known_dstasns_list
0, 16000
```

Possible error responses can be:

```
208 AUTH_ERROR login:passwd:newchecksum
known_dstasns_list
```

Or:

```
208 DOWN_ERROR login:passwd:newchecksum
known_dstasns_list
```

List a Filter's Attributes

```
209 GET login:passwd:checksum
filter_attributes:allow-www
```

An expected response is:

```
209 GET_RESPONSE login:passwd:newchecksum
filter_attributes:allow-www
Filter allow-www
Permit Srcport   80
Permit DstPort 80
Permit Dstaddr 0.0.0.0 255.255.255.255
```

A possible error response can be (excluding AUTH_ERROR and DOWN_ERROR):

```
209 GET_ERROR login:passwd:newchecksum
filter_attributes:allow-www
error:filter ID not found
```

List a Thread's Attributes

```
301 GET login:passwd:checksum
thread_attributes:HMAT
```

An expected response is:

```
301 GET_RESPONSE login:passwd:newchecksum
thread_attributes:HMAT
Thread HMAT
Aggregation HostMatrix
Period 20
Port 9991
State Active
DataSetPath /opt/CSCOnfc/Data
   Compression No
   Binary No
   MaxUsage 500
```

A possible error response can be (excluding AUTH_ERROR and DOWN_ERROR):

```
301 GET_ERROR login:passwd:newchecksum
thread_attributes:allow-www
error:thread ID not found
```

List a Protocol's Attributes

```
302 GET login:passwd:checksum
protocol_attributes:TELNET
```

An expected response is:

```
302 GET_RESPONSE login:passwd:newchecksum
protocol_attributes:TELNET
Protocol TELNET
Srcport 23 OR
Dstport 23
prot 6
```

A possible error response can be (excluding AUTH_ERROR and DOWN_ERROR):

```
302 GET_ERROR login:passwd:newchecksum
protocol_attributes:TELNET
error:protocolid not found
```

SET and SET_RESPONSE and SET_ERROR

In this section, AUTH_ERROR and DOWN_ERROR are not shown.

Start FlowCollector

```
500 SET login:passwd:checksum
start_collector
```

An expected sample response is:

```
500 SET_RESPONSE login:passwd:newchecksum
start_collector
```

A possible error response can be:

```
500 SET_ERROR login:passwd:newchecksum
start_collector
error:running
```

Note You can optionally specify an argument to the **start_collector** command to indicate how long FlowCollector should run (for example, 60 minutes).

Stop FlowCollector

```
501 SET login:passwd:checksum
stop_collector
```

An expected sample response is:

```
501 SET_RESPONSE login:passwd:newchecksum
stop_collector
info:stopped at Tue Jun 23 23:10:02 1998
```

A possible error response can be:

```
501 SET_ERROR login:passwd:newchecksum
stop_collector
error:not running
info:stopped at Tue Jun 23 23:10:02 1998
```

Request to Write FlowCollector's In-Memory Configuration into the Log File

This section refers to requests that are local to the FlowCollector box.

```
501 SET login:passwd:checksum
dump_config
```

An expected sample response is:

```
501 SET_RESPONSE login:passwd:newchecksum
dump_configr
```

Add a Filter

```
502 SET login:passwd:checksum
add_filter:DENY-ICMP
Deny Prot 2
Permit Srcaddr 0.0.0.0 255.255.255.255
```

An expected sample response is:

```
502 SET_RESPONSE login:passwd:newchecksum
add_filter:DENY-ICMP
Deny Prot 2
Permit Srcaddr 0.0.0.0 255.255.255.255
```

Possible error responses can be:

```
502 SET_ERROR login:passwd:newchecksum
add_filter:DENY-ICMP
error:duplicate filterid
```

Or:

```
502 SYNTAX_ERROR login:passwd:newchecksum
add_filter:DENY-ICMP
error:syntax error
```

Drop a Filter

```
503 SET login:passwd:checksum
drop_filter:DENY-ICMP
```

An expected sample response is:

```
503 SET_RESPONSE login:passwd:newchecksum
drop_filter:DENY-ICMP
info:dropped Filter DENY-ICMP
```

A possible error response can be:

```
503 SET_ERROR login:passwd:newchecksum
drop_filter:DENY-ICMP
error:filter ID not found
```

Add a Thread

```
504 SET login:passwd:checksum
add_thread:SRCNODE
Aggregation SourceNode
Filter DENY-ICMP
Period 15
Port 9991
State Active
DataSetPath /opt/CSCOnfc/Data
   Compression No
   Binary No
   MaxUsage 500
```

An expected sample response is:

```
504 SET_RESPONSE login:passwd:newchecksum
add_thread:SRCNODE
```

Possible error responses (only the nonidentical portions are shown) can be:

```
error:duplicate threadid
```

Or:

```
error:max active threads limit reached
```

Or:

```
error:port, datasetpath and aggregation conflict
```

Or:

```
error:syntax error
```

Drop a Thread

```
505 SET login:passwd:checksum
drop_thread:SRCNODE
```

An expected sample response is:

```
505 SET_RESPONSE login:passwd:newchecksum
drop_thread:SRCNODE
info:dropped Thread SRCNODE
```

A possible error response can be:

```
505 SET_ERROR login:passwd:newchecksum
drop_thread:SRCNODE
error:thread ID not found
Modify a thread:
506 SET login:passwd:checksum
thread_attributes:SRCNODE
Period 15
Port 9991
State Active
DataSetPath /opt/CSCOnfc/Data
   Compression No
   Binary No
   MaxUsage 500
```

An expected sample response is:

```
506 SET_RESPONSE login:passwd:newchecksum
thread_attributes:SRCNODE
Thread SRCNODE
Period 15
Port 9991
State Active
DataSetPath /opt/CSCOnfc/Data
   Compression No
   Binary No
   MaxUsage 500
```

Possible error responses (only the nonidentical portions are shown) can be:

```
error:thread ID not found
```

Or:

```
error:max active threads limit reached
```

Or:

```
error:port, datasetpath and aggregation conflict
```

Or:

```
error:syntax error
Add a Protocol
507 SET login:passwd:checksum
add_protocol:UDP-NNTP
Srcport 119 OR Dstport 119
Prot 17
```

An expected sample response is:

```
507 SET_RESPONSE login:passwd:newchecksum
add_protocol:UDP-NNTP
info:dropped Protocol UDP-NNTP
```

Possible error responses can be:

```
507 SET_ERROR login:passwd:newchecksum
add_protocol:UDP-NNTP
error:duplicate protocol ID
```

Or:

```
507 SET_ERROR login:passwd:newchecksum
add_protocol:UDP-NNTP
error:syntax error
```

Drop a Protocol

```
508 SET login:passwd:checksum
drop_protocol:DENY-WWW
```

An expected sample response is:

```
508 SET_RESPONSE login:passwd:newchecksum
drop_protocol:WWW-SERVER
```

Possible error responses can be:

```
508 SET_ERROR login:passwd:newchecksum
drop_protocol:WWW-SERVER
error:protocolid not foundr

Or:
508 SET_ERROR login:passwd:newchecksum
drop_protocol:WWW-SERVER
error:syntax error
```

Unsolicited Event Notifications (UENs), or Trap Messages

Disk Consumption Reaches a Certain Limit

```
801 EVENT checksum
disk_consumption: /opt/CSCOnfc
info: 80%
802 EVENT checksum
disk_consumption: /opt/CSCOnfc
info: 100%
```

Unsolicited Packet Is Received

```
802 EVENT checksum
rcvd_unsolicited_packet:171.71.34.79
```

Active Threads Limit Is Reached

```
800 EVENT checksum
max_active_threads_limit
```

Incompatible Packet with an Aggregation Scheme

```
810 EVENT checksum
incompatible_pkt_and_aggregation
info: V1 packet for ASMatrix
```

Could Not Write a Data File

```
815 EVENT checksum
datafile_not_written:/opt/CSCOnfc/Data/R1/1998_10_12/r1.1215
info: insufficient space in /opt/CSCOnfc
```