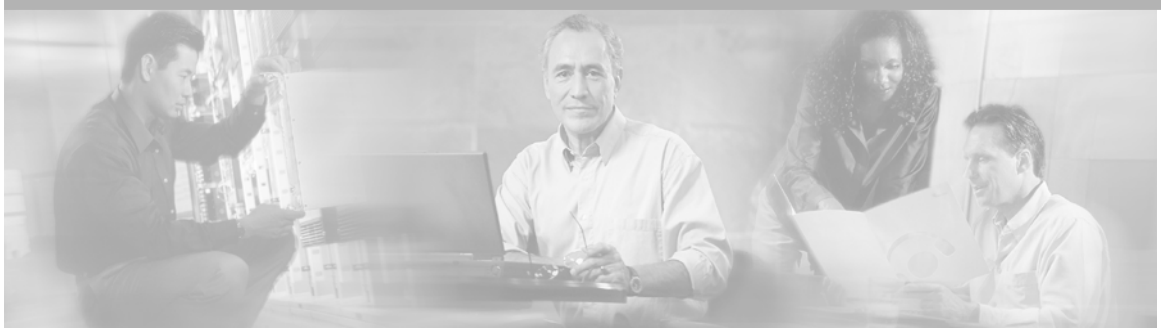


Cisco CNS NetFlow Collection Engine User Guide, 5.0.3

October, 2005

Corporate Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 526-4100

Customer Order Number: N/A
Text Part Number: OL-6899-01

Cisco CNS NetFlow Collection Engine User Guide, 5.0.3

About This Guide ix

Objective ix

Audience ix

How This Guide Is Organized x

Command Syntax Conventions xi

Obtaining Documentation xi

World Wide Web xi

Ordering Documentation xi

Documentation Feedback xii

Obtaining Technical Assistance xii

Cisco.com xii

Technical Assistance Center xii

Cisco Technical Support Web Site xiii

Cisco TAC Escalation Center xiii

CHAPTER 1 **Overview** 1-1

What Are NetFlow Services? 1-1

NetFlow Services Device and IOS Release Support 1-2

NetFlow Data Export 1-2

How and When Flow Statistics Are Exported 1-2

NetFlow Data Export Formats 1-3

What Is CNS NetFlow Collection Engine? 1-4

CNS NetFlow Collection Engine Architectural Overview 1-5

Collector 1-6

Web-Based User Interface 1-6

CNS/XML Interface 1-7

Report Generator 1-7

BGP Peer 1-7

CHAPTER 2 **Using the CNS NetFlow Collection Engine User Interface** 2-1

Starting the CNS NetFlow Collection Engine User Interface 2-1

Customizing the CNS NetFlow Collection Engine Interface 2-2

Web-based User Interface Authentication 2-2

Advanced Configuration 2-3

Using the CNS NetFlow Collection Engine User Interface 2-3
Login Screen 2-3
Navigation 2-4
Configuration 2-6
Global 2-6
Fields 2-7
Key Builders 2-8
Address Range Map 2-9
BGP Attribute 2-9
Bit Field 2-10
Interface SNMP Name 2-10
Integer 2-11
Integer Range Map 2-11
Internet Address 2-11
Mask Field Internet Address 2-12
Masked Internet Address 2-12
Multi-Field Map 2-12
Option Data 2-13
String 2-14
Value Builders 2-14
Active Time 2-15
End Time 2-15
Flow Count 2-15
Max Burst Rate 2-16
Rate 2-16
Start Time 2-16
Sum 2-16
Aggregators 2-17
Aggregation Schemes 2-18
Filters 2-19
NetFlow Export Source Groups 2-21
NetFlow Export Source Access List 2-22
BGP Peer 2-23
Advanced 2-24
Reports 2-25
Custom Reports 2-26
Common Reports 2-30
Scheduled Reports 2-30
Configuring Scheduled Reports 2-30
Displaying Scheduled Reports 2-33

Reporting Features 2-34
Sorting and Graphing 2-35
Export and Print 2-36
Filter 2-36
Drill Down 2-37
Status 2-37
Control 2-37
Statistics 2-38
Port Statistics 2-38
Source Statistics 2-38
Logs 2-39
Troubleshooting 2-40

CHAPTER 3 **Understanding the CNS NetFlow Collection Engine Data File Format** 3-1
Data File Directory Structure 3-1
Default Data File Directory Structure 3-1
Options Data Directory File Structure 3-3
Data Filenames 3-4
Data File Format 3-4
Backwards-Compatible Header 3-5
XML Header Format 3-7
Partial Data Files 3-8
Options Data File Format 3-9
Using the filesready File to Track Data Files 3-9
Options Data Filesready File 3-10

CHAPTER 4 **Customizing the CNS NetFlow Collection Engine** 4-1
CNS NetFlow Collection Engine Configuration and Resource Files 4-1
XML Configuration and Schema 4-1
CNS NetFlow Collection Engine Data Collection and Aggregation 4-2
Overview of CNS NetFlow Collection Engine 5.0 Configuration Information 4-2
Fields 4-4
Keys and Values 4-5
Key Builder Types 4-5
Value Builder Types 4-12
Key and Value Builder Data Types 4-13
Creating an Aggregation Scheme 4-15
Creating an Aggregator 4-16

Creating a Filter 4-19
Creating a Map 4-20
Creating a Multi-Field Map 4-21
Creating an Option Data Map 4-22
Creating Source Groups 4-24
Creating Access Lists 4-24
Global Settings 4-25
Setting the Time Zone 4-26
Tuning Memory Usage 4-26
Managing Disk Space 4-28
Filters 4-28
Aggregation 4-28
Data File and Disk Space Options 4-28
Monitoring Disk Usage 4-29
NetFlow Collection Engine Logger Configuration 4-29
Rolling File Option for NFC Logs Files 4-29
Daily Rolling File Option for NFC Log Files 4-30
Configuring the Logger from the Command Line 4-31

CHAPTER 5 **CNS NetFlow Collection Engine Advanced Features** 5-1
Process Watcher 5-1
Configuration 5-2
Event Service 5-3
Configuration 5-3
Message Format 5-4
Sample Output 5-4
Report Generator 5-4
BGP Peer 5-8
Starting and Stopping the BGP Peer 5-8
Configuration 5-8
Interface Name Support 5-10

APPENDIX A **Troubleshooting CNS NetFlow Collection Engine** A-1
Using the nfcollector status Command A-1
Using the show-tech Command to Capture Troubleshooting Information A-2
CNS NetFlow Collection Engine Tools and Utilities A-2
fdcount Utility A-2
ndeget Utility A-3

showpacketlog Utility A-3
get_bgp_rib Utility A-3
fdget Utility A-4
fdplayback Utility A-4
nfc_gunzip Utility A-5
nfc_bin_to_ascii Utility A-5
Solving CNS NetFlow Collection Engine Problems A-5
Web-based UI Troubleshooting Tips A-9

APPENDIX B **NetFlow Export Datagram Formats** B-1
Versions 1, 5, 7 and 8 B-1
Version 9 B-14

APPENDIX C **CNS NetFlow Collection Engine Binary Data File Format** C-1

APPENDIX D **Logging** D-1
Configuration D-1

APPENDIX E **CNS NetFlow Collection Engine CNS/XML Interface** E-1
Configuration E-1
Usage E-2
Message Format E-2
Supported XML Requests E-3
Response Status E-4

APPENDIX F **Sample Work Flow** F-1

APPENDIX G **CNS NetFlow Collection Engine Migration Tools** G-1
CNS NetFlow Collection Engine Configuration Migration G-1
Using the Configuration Migration Tool G-1
CNS NetFlow Collection Engine Data Migration G-2
Using the Data Migration Tool G-2

INDEX

About This Guide

Objective

The *Cisco CNS NetFlow Collection Engine User Guide, Release 5.0.3* describes the CNS NetFlow Collection Engine application, which is used with the NetFlow services data export feature on Cisco routers and Catalyst 5000 and 6000 series switches. This document also describes the system requirements that must be met to install the CNS NetFlow Collection Engine product, as well as, how to install, start, and configure CNS NetFlow Collection Engine.

NetFlow services consist of high-performance IP switching features that capture a rich set of traffic statistics exported from routers and switches while they perform their switching function. CNS NetFlow Collection Engine provides fast, scalable, and economical data collection from multiple export devices exporting NetFlow data records.

Prior to reading this manual, you should read the *Release Notes for Cisco CNS NetFlow Collection Engine Release 5.0* document. These release notes provide information about known software and documentation problems and any last minute information about the CNS NetFlow Collection Engine software not available when this guide was produced.

In previous releases, this product was referred to as Cisco NetFlow FlowCollector (NFC).

Audience

This guide is intended primarily for individuals with network and system administration skills. You should have a basic understanding of network design, operation, and terminology, as well as familiarity with your own network configurations. You also must have a basic familiarity with Web browsers, Hewlett Packard's HP-UX, or Sun Microsystem's Solaris Operating System.

How This Guide Is Organized

This guide is organized as follows:

Chapter 1, "Overview," describes the CNS NetFlow Collection Engine application.

Chapter 2, "Using the CNS NetFlow Collection Engine User Interface," describes how to use the CNS NetFlow Collection Engine user interface (NFUI) to review application statistics and resource definitions.

Chapter 3, "Understanding the CNS NetFlow Collection Engine Data File Format," describes how to interpret the data collected and saved in CNS NetFlow Collection Engine data files.

Chapter 4, "Customizing the CNS NetFlow Collection Engine," describes how to customize CNS NetFlow Collection Engine operations.

Chapter 5, "CNS NetFlow Collection Engine Advanced Features," describes CNS NetFlow Collection Engine advanced features such as process watcher, event service, report generator, and BGP peer.

Appendix A, "Troubleshooting CNS NetFlow Collection Engine," provides helpful information and procedures in case you encounter problems while using CNS NetFlow Collection Engine.

Appendix B, "NetFlow Export Datagram Formats," describes how NetFlow exports flow information in UDP datagrams in one of four formats.

Appendix C, "CNS NetFlow Collection Engine Binary Data File Format," describes the format of the CNS NetFlow Collection Engine binary data file.

Appendix D, "Logging," describes the CNS NetFlow Collection Engine logging functions.

Appendix E, "CNS NetFlow Collection Engine CNS/XML Interface," describes the CNS NetFlow Collection Engine CNS/Xtensible Markup Language (XML) interface.

Appendix F, "Sample Work Flow," contains a sample work flow to use as a reference.

Appendix G, "CNS NetFlow Collection Engine Migration Tools," describes tools to facilitate migrating configurations and data from previous versions of CNS NetFlow Collection Engine

An Index is also provided.

Command Syntax Conventions

Table 1 describes the syntax used with the commands in this document.

Table 1 Command Syntax Guide

Convention	Description
boldface	Commands and keywords.
italic	Command input that is supplied by you.
[ ]	Keywords or arguments that appear within square brackets are optional.
{ x | x | x }	A choice of keywords (represented by x) appears in braces separated by vertical bars. You must select one.
^ or Ctrl	Represent the key labeled *Control*. For example, when you read ^*D* or *Ctrl-D*, you should hold down the Control key while you press the D key.
`screen font`	Examples of information displayed on the screen.
boldface screen font	Examples of information that you must enter.
< >	Nonprinting characters, such as passwords, appear in angled brackets.
[ ]	Default responses to system prompts appear in square brackets.

Obtaining Documentation

The following sections explain how to obtain documentation from Cisco Systems.

World Wide Web

You can access the most current Cisco documentation on the World Wide Web at the following URL:

http://www.cisco.com

Translated documentation is available at the following URL:

http://www.cisco.com/public/countries_languages.shtml

Ordering Documentation

Cisco documentation is available in the following ways:

- Registered Cisco Direct Customers can order Cisco product documentation from the Networking Products MarketPlace:

  http://www.cisco.com/cgi-bin/order/order_root.pl

- Registered Cisco.com users can order the Documentation CD-ROM through the online Subscription Store:

  http://www.cisco.com/go/subscription

- Nonregistered Cisco.com users can order documentation through a local account representative by calling Cisco corporate headquarters (California, USA) at 408 526-7208 or, elsewhere in North America, by calling 800 553-NETS (6387).

Documentation Feedback

If you are reading Cisco product documentation on Cisco.com, you can submit technical comments electronically. Click **Leave Feedback** at the bottom of the Cisco Documentation home page. After you complete the form, print it out and fax it to Cisco at 408 527-0730.

You can e-mail your comments to bug-doc@cisco.com.

To submit your comments by mail, use the response card behind the front cover of your document, or write to the following address:

Cisco Systems
Attn: Document Resource Connection
170 West Tasman Drive
San Jose, CA 95134-9883

We appreciate your comments.

Obtaining Technical Assistance

Cisco provides Cisco.com as a starting point for all technical assistance. Customers and partners can obtain documentation, troubleshooting tips, and sample configurations from online tools by using the Cisco Technical Assistance Center (TAC) Web Site. Cisco.com registered users have complete access to the technical support resources on the Cisco Technical Support Web Site.

Cisco.com

Cisco.com is the foundation of a suite of interactive, networked services that provides immediate, open access to Cisco information, networking solutions, services, programs, and resources at any time, from anywhere in the world.

Cisco.com is a highly integrated Internet application and a powerful, easy-to-use tool that provides a broad range of features and services to help you to

- Streamline business processes and improve productivity
- Resolve technical issues with online support
- Download and test software packages
- Order Cisco learning materials and merchandise
- Register for online skill assessment, training, and certification programs

You can self-register on Cisco.com to obtain customized information and service. To access Cisco.com, go to the following URL:

http://www.cisco.com

Technical Assistance Center

The Cisco TAC is available to all customers who need technical assistance with a Cisco product, technology, or solution. Two types of support are available through the Cisco TAC: the Cisco Technical Support Web Site and the Cisco TAC Escalation Center.

Inquiries to Cisco TAC are categorized according to the urgency of the issue:

- Priority level 4 (P4)—You need information or assistance concerning Cisco product capabilities, product installation, or basic product configuration.
- Priority level 3 (P3)—Your network performance is degraded. Network functionality is noticeably impaired, but most business operations continue.
- Priority level 2 (P2)—Your production network is severely degraded, affecting significant aspects of business operations. No workaround is available.
- Priority level 1 (P1)—Your production network is down, and a critical impact to business operations will occur if service is not restored quickly. No workaround is available.

Which Cisco TAC resource you choose is based on the priority of the problem and the conditions of service contracts, when applicable.

Cisco Technical Support Web Site

The Cisco Technical Support Web Site allows you to resolve P3 and P4 issues yourself, saving both cost and time. The site provides around-the-clock access to online tools, knowledge bases, and software. To access the Cisco Technical Support Web Site, go to the following URL:

http://www.cisco.com/tac

All customers, partners, and resellers who have a valid Cisco services contract have complete access to the technical support resources on the Cisco Technical Support Web Site. The Cisco Technical Support Web Site requires a Cisco.com login ID and password. If you have a valid service contract but do not have a login ID or password, go to the following URL to register:

http://www.cisco.com/register/

If you cannot resolve your technical issues by using the Cisco Technical Support Web Site, and you are a Cisco.com registered user, you can open a case online by using the TAC Case Open tool at the following URL:

http://www.cisco.com/tac/caseopen

If you have Internet access, it is recommended that you open P3 and P4 cases through the Cisco Technical Support Web Site.

Cisco TAC Escalation Center

The Cisco TAC Escalation Center addresses issues that are classified as priority level 1 or priority level 2; these classifications are assigned when severe network degradation significantly impacts business operations. When you contact the TAC Escalation Center with a P1 or P2 problem, a Cisco TAC engineer will automatically open a case.

To obtain a directory of toll-free Cisco TAC telephone numbers for your country, go to the following URL:

http://www.cisco.com/warp/public/687/Directory/DirTAC.shtml

Before calling, please check with your network operations center to determine the level of Cisco support services to which your company is entitled; for example, SMARTnet, SMARTnet Onsite, or Network Supported Accounts (NSA). In addition, please have available your service agreement number and your product serial number.

Overview

This chapter describes the CNS NetFlow Collection Engine application, which is used with the NetFlow services data export feature on Cisco routers and Catalyst 5000 and 6000 series switches.

This chapter includes the following sections:

- What Are NetFlow Services?
- What Is CNS NetFlow Collection Engine?
- CNS NetFlow Collection Engine Architectural Overview

What Are NetFlow Services?

NetFlow services consist of high-performance IP switching features that capture a rich set of traffic statistics exported from routers and switches while they perform their switching functions. The exported NetFlow data consists of traffic flows, which are unidirectional sequences of packets between a particular source device and destination device that share the same protocol and transport-layer information. The captured traffic statistics can be used for a wide variety of purposes, such as network analysis and planning, network management, accounting, billing, and data mining.

Because of their unidirectional nature, flows from a client to a server are differentiated from flows from the server to the client. Flows are also differentiated on the basis of protocol. For example, Hypertext Transfer Protocol (HTTP) Web packets from a particular source host to a particular destination host constitute a separate flow from File Transfer Protocol (FTP) file transfer packets between the same pair of hosts.

Routers and switches identify flows by looking for the following fields within IP packets:

- Source IP address
- Destination IP address
- Source port number
- Destination port number
- Protocol type
- Type of service (ToS)
- Input interface

Catalyst 5000 series switches can identify flows by looking at a subset of these fields. For example, they can identify flows by source and destination address only.

Note For Catalyst 5000 series switches, the analog to NetFlow services is integrated Multilayer Switching (MLS) management. Included are products, utilities, and partner applications designed to gather flow statistics, export the statistics, and collect and perform data reduction on the exported statistics. MLS management then forwards them to consumer applications for traffic monitoring, planning, and accounting.

NetFlow Services Device and IOS Release Support

You can find the most up-to-date information available to help you determine the compatibility among different Cisco hardware platforms, Cisco IOS software releases, and supported NetFlow data export versions at the following URL:

http://tools.cisco.com/ITDIT/CFN/Dispatch?SearchText=Netflow&act=featSelect&rnFeatId=null&featStartsWith=&task=TextSearch&altrole=

Note Except for descriptions requiring references to specific router or switch platforms, the remainder of this chapter and the remaining chapters of this guide use the term export device instead of the terms router and switch.

NetFlow Data Export

NetFlow data export makes NetFlow traffic statistics available for purposes of network planning, billing, and so on. An export device configured for NetFlow data export maintains a flow cache used to capture flow-based traffic statistics. Traffic statistics for each active flow are maintained in the cache and are updated when packets within each flow are switched. Periodically, summary traffic statistics for all expired flows are exported from the export device by means of User Datagram Protocol (UDP) datagrams, which CNS NetFlow Collection Engine receives and processes.

How and When Flow Statistics Are Exported

NetFlow data exported from the export device contains NetFlow statistics for the flow cache entries that have expired since the last export. Flow cache entries expire and are flushed from the cache when one of the following conditions occurs:

- The transport protocol indicates that the connection is completed (TCP FIN) plus a small delay to allow for the completion of the FIN acknowledgment handshaking.
- Traffic inactivity exceeds 15 seconds.

For flows that remain continuously active, flow cache entries currently expire every 30 minutes to ensure periodic reporting of active flows.

NetFlow data export packets are sent to a user-specified destination, such as the workstation running CNS NetFlow Collection Engine, either when the number of recently expired flows reaches a predetermined maximum, or every second-whichever occurs first. For:

- Version 1 datagrams, up to 24 flows can be sent in a single UDP datagram of approximately 1200 bytes.

- Version 5 datagrams, up to 30 flows can be sent in a single UDP datagram of approximately 1500 bytes.
- Version 7 datagrams, up to 27 flows can be sent in a single UDP datagram of approximately 1500 bytes.
- Version 8 datagrams, the number of flows sent in a single UDP datagram varies by aggregation scheme.
- Version 9 datagrams, the number of flows is variable, and depends on the number and size of fields defined in one or more templates.

See Appendix B, "NetFlow Export Datagram Formats," for details on all versions of the NetFlow data export format.

NetFlow Data Export Formats

NetFlow exports flow information in UDP datagrams in one of five formats: Version 1 (V1), Version 5 (V5), Version 7 (V7), Version 8 (V8), or Version 9 (V9).

Version 1 is the original format supported in the initial NetFlow releases. Version 5 is an enhancement that adds Border Gateway Protocol (BGP) autonomous system information and flow sequence numbers. Version 7 is an enhancement that exclusively supports Cisco Catalyst 5000 series switches equipped with a NetFlow feature card (NFFC). V7 is not compatible with Cisco routers. Version 8 is an enhancement that adds router-based aggregation schemes. Version 9 is an enhancement to support different technologies such as Multicast, Internet Protocol Security (IPSec), and Multi Protocol Label Switching (MPLS). CNS NetFlow Collection Engine Release 5.0 can collect, filter, and aggregate Version 9 data in the same way it does for NetFlow Data Export Versions 1 through 8.

Versions 2, 3, 4, and 6 are not supported by CNS NetFlow Collection Engine. For more information on the distinctions among the NetFlow data export formats, see Appendix B, "NetFlow Export Datagram Formats."

The following types of information are part of the detailed traffic statistics:

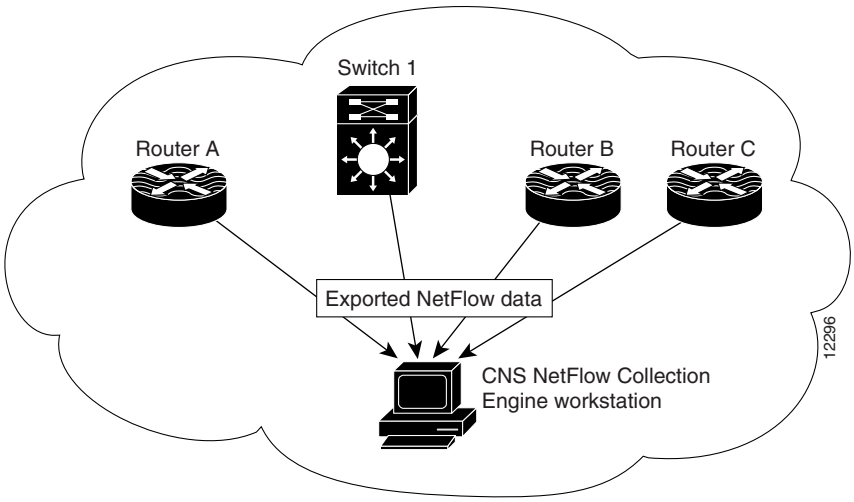
- Source and destination IP addresses
- Next hop address
- Input and output interface numbers
- Number of packets in the flow
- Total bytes (octets) in the flow
- First and last time stamps of packets that were switched as part of this flow
- Source and destination port numbers
- Protocol
- Type of service (ToS)
- Source and destination autonomous system (AS) numbers, either origin or peer (present in V5 and select V8 datagrams)
- Source and destination prefix mask bits (present in V5, V7, and V8 datagrams)
- Shortcut router IP address (present in V7 on Cisco Catalyst 5000 series switches only).

Caution Throughout this publication there are numerous examples of CNS NetFlow Collection Engine input commands and output results. Included are examples of IP addresses. Be aware that IP address examples are not usable IP addresses. The examples do not represent real-life configurations.

What Is CNS NetFlow Collection Engine?

CNS NetFlow Collection Engine provides fast, scalable, and economical data collection from multiple export devices exporting NetFlow data records. Figure 1-1 shows an example of a typical NetFlow data export scheme. In it, various export devices send export data to user-specified CNS NetFlow Collection Engine UDP ports.

Figure 1-1 CNS NetFlow Collection Engine Overview



Each of the export devices in this example is configured for NetFlow data export. Part of the configuration information for each export device includes the IP address and the UDP port number (a logical port designator) that identify CNS NetFlow Collection Engine as the receiver of flows from this export device. The UDP port number is a user-configurable designator: you can configure CNS NetFlow Collection Engine to listen for flows on a number of different UDP ports, and then configure your export devices so that each device exports flows to a dedicated UDP port, or have a number of devices export flows to the same, shared UDP port.

After you configure and start CNS NetFlow Collection Engine, it listens to the user-specified UDP ports for exported flows from the export devices you have configured for NetFlow data export.

CNS NetFlow Collection Engine performs the following functions:

- NetFlow data collection from multiple export devices
- Reduction in data volume through filtering and aggregation
- Hierarchical data storage (helps client applications retrieve data)
- File system space management

CNS NetFlow Collection Engine collects and summarizes (aggregates) data into data files based on user-defined criteria specified in a CNS NetFlow Collection Engine *aggregator*. An *aggregator* is an aggregation task defined by a set of user-configurable attributes that specify how CNS NetFlow Collection Engine summarizes the traffic flows that are received. Two important aggregator attributes are:

- Aggregation schemes – defines the subset of data of interest in a traffic flow, as well as which statistics are kept
- Filter – criteria for accepting or rejecting flows that are aggregated or summarized

CNS NetFlow Collection Engine provides a set of predefined aggregation schemes to help you collect NetFlow export data and summarize the data (that is, aggregate the flows). You can choose one or more of these aggregation schemes to customize CNS NetFlow Collection Engine for your operating context. Moreover, in Release 5.0 you can modify any of the predefined aggregation schemes or define your own aggregation schemes based on them. You can also use filters with aggregation schemes to include or exclude certain types of NetFlow data.

For more information about threads, aggregation schemes, and filters, see Chapter 4, "Customizing the CNS NetFlow Collection Engine."

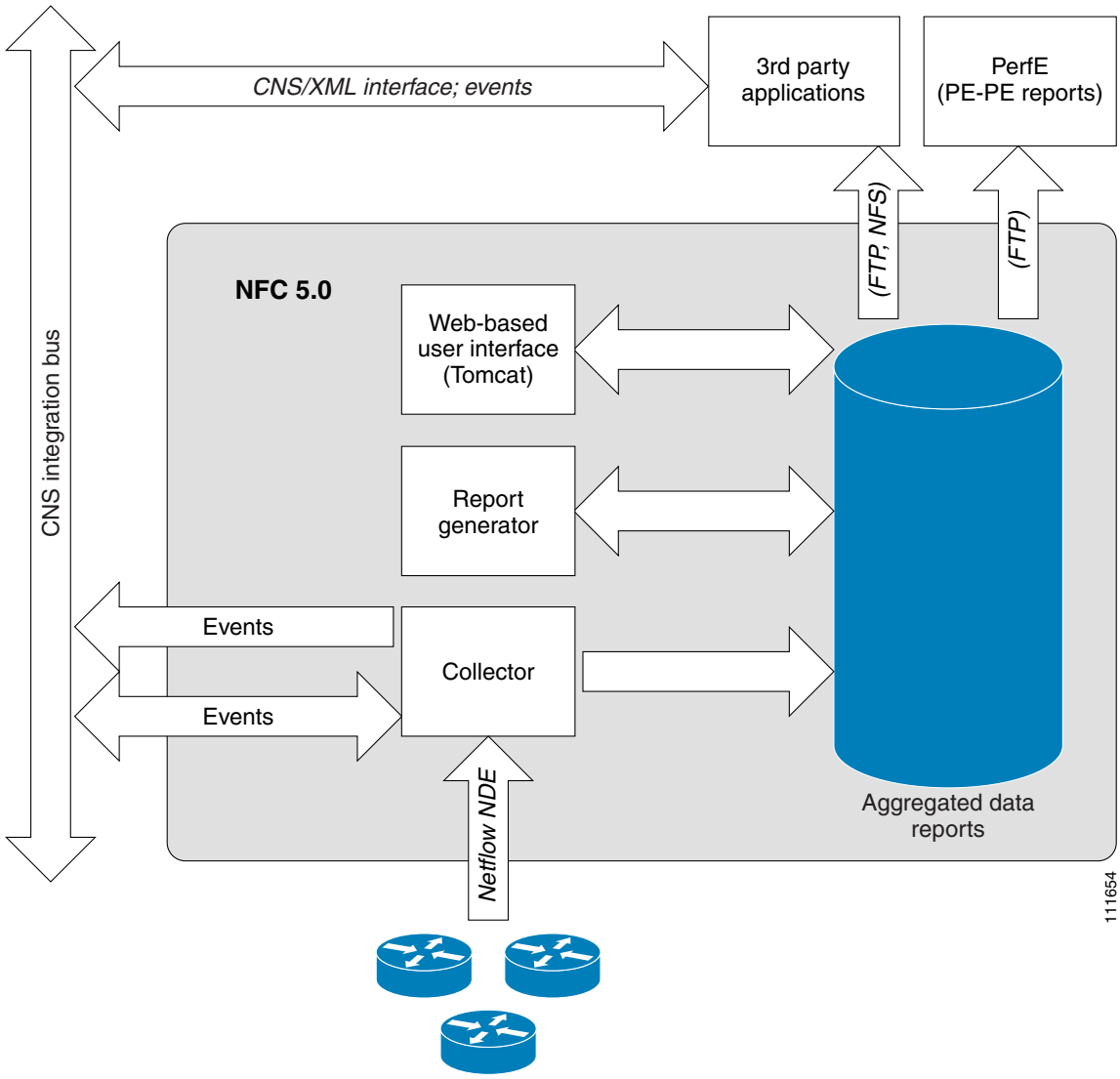
CNS NetFlow Collection Engine Architectural Overview

CNS NetFlow Collection Engine consists of the following components:

- Collector
- Web-based User Interface (UI)
- CNS/XML Interface
- Reporting engine
- Border Gateway Protocol (BGP) Peer

These subsystems work together to provide CNS NetFlow Collection Engine functionality, including data collection, the user interface, configuration and control, and reporting. They also allow custom client applications to interface with CNS NetFlow Collection Engine. See Figure 1-2 for a graphical representation of the CNS NetFlow Collection Engine system architecture.

Figure 1-2 CNS NetFlow Collection Engine System Architecture



Collector

The Collector subsystem collects NetFlow data, aggregates (or summarizes) that data, and filters specified data from supported Cisco routers and switches. Output is stored in files that are organized in an easy-to-use directory structure.

Web-Based User Interface

The Web-Based User Interface is provided for configuration, control, status, and reporting.

CNS/XML Interface

The CNS/XML Interface is used to send and receive configuration/control requests and responses, and unsolicited event notifications. The CNS/XML interface uses the CNS Integration Bus to communicate with clients.

Report Generator

The Report Generator produces hourly and daily reports based on Collector output files by performing further aggregation of the records in these files based on criteria selected by the user.

BGP Peer

A passive BGP peer is provided for supplementing CNS NetFlow Collection Engine output with BGP attributes.

Using the CNS NetFlow Collection Engine User Interface

CNS NetFlow Collection Engine, Release 5.0 has a new web-based user interface (UI) for configuration, control, and reporting. Each collector instance has a web server that the user can start to enable the web-based UI.

This chapter includes the following sections:

- Starting the CNS NetFlow Collection Engine User Interface, page 2-1
- Customizing the CNS NetFlow Collection Engine Interface, page 2-2.
- Using the CNS NetFlow Collection Engine User Interface, page 2-3
- Configuration, page 2-6
- Reports, page 2-25
- Status, page 2-37
- Troubleshooting, page 2-40

Starting the CNS NetFlow Collection Engine User Interface

To start the CNS NetFlow Collection Engine User Interface, do the following:

Step 1 To run CNS NetFlow Collection Engine, log in as the user specified during installation.

Step 2 Enter the following command:

```
/opt/CSCOnfc/bin/nfcollector start all
```

Step 3 From a web browser using the UI, enter:

```
http://<nfc-hostname>:8080/nfc in a web browser to begin using the UI.
```

Note The web-based UI only works with the collector located on the same machine. To access a different instance of CNS NetFlow Collection Engine you must start that collector's web server and access it through the corresponding URL.

Customizing the CNS NetFlow Collection Engine Interface

Table 2-1 describes the settings that can be customized for the CNS NetFlow Collection Engine web-based UI.

Table 2-1 CNS NetFlow Collection Engine User Interface Settings

Setting	Description	Default Value	File
intfc- password	Digest password for the CNS/XML interface. Stored as a parameter to the InitServlet in the servlet configuration file. This setting must match the md5-password value of the CNS/XML interface.	password	NFC_DIR/tomcat/webapps/nfc/ WEB- INF/web.xml
port	The port on which the web server listens for HTTP connections. Stored as an attribute in the web server configuration file.	8080	NFC_DIR/tomcat/conf/server.xml
session-timeout	A session is started once a user logs in to the web-based UI. This timeout indicates the duration of inactivity allowed before a session expires and the user is automatically logged out. Add:`<session-config><session-timeout>30</session-timeout></session-config>` after all `<servlet-mappings>`.	30 minutes	NFC_DIR/tomcat/webapps/nfc/ WEB-INF/web.xml

Web-based User Interface Authentication

In CNS NetFlow Collection Engine releases prior to 5.0.3, authentication information for a single web interface user was configured in the file **$NFC_DIR/tomcat/webapps/nfc/WEB- INF/web.xml**. Starting in Release 5.0.3, authentication is performed via a JAAS (Java Authentication and Authorization Service) plug-in. Authentication information is configured in the file **$NFC_DIR/config/auth.config**. Two sample strategies are provided: simple authentication and UNIX authentication.

Simple authentication is enabled by default. To enable simple authentication, copy **auth.config.simple** in **$NFC_DIR/config** to **auth.config**, and optionally change the user name and password:

```
NFC {
    com.cisco.nfc.collector.web.auth.SimpleLoginModule required nfc-user="nfcuser"
nfc-password="nfcuser";
};
```

As shown above, when simple authentication is enabled, the default user name and password for accessing the web-based user interface is **nfcuser**.

To enable UNIX authentication, copy **auth.config.unix** in **$NFC_DIR/config** to **auth.config** and set the account information there as appropriate for your system:

```
NFC {
    com.cisco.nfc.collector.web.auth.UnixUserLoginModule required debug="false"
libs="${NFC_DIR}/lib/UnixAccountInfo.so" nfc-users="nfcuser,root";
};
```

When UNIX authentication is enabled, the *nfc-users* property contains a comma-separated list of system users that are authorized to access the web-based user interface, *nfcuser* and *root* in the example above. Similarly, the property *nfc-group* can be added that contains the name of a system group whose users are authorized to access the web-based user interface. Note that because the UNIX authentication plug-in uses standard UNIX programming APIs to access account information, it is the user's responsibility that this account information (specifically the crypted password in /etc/passwd, /etc/shadow, NIS+, etc.) is accessible through these calls. Consult with your system administrator about what is expected for this; the administration of UNIX account information is beyond the scope of this document.

Advanced Configuration

It is possible to configure the CNS NetFlow Collection Engine web server to use SSL for encrypting connections. See **http://jakarta.apache.org/tomcat/tomcat-3.3-doc/tomcat-ssl-howto.html** for instructions.

CNS Integration Bus settings can be set with subject, service, network, and daemon parameters to the InitServlet in the servlet configuration file.

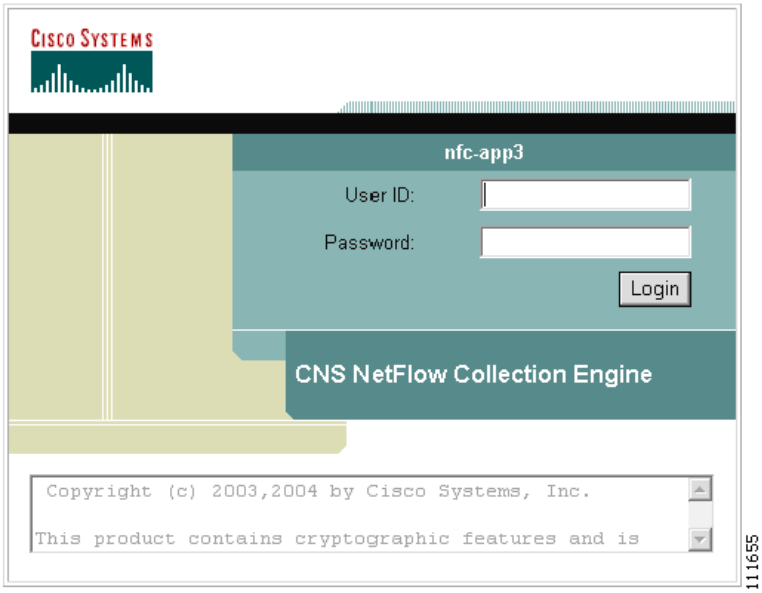
Using the CNS NetFlow Collection Engine User Interface

The following sections describes using the CNS NetFlow Collection Engine User Interface.

Login Screen

The first page that loads is always the login screen as shown in Figure 2-1. For security purposes, to use the web-based UI you must authenticate yourself with a user ID and password. These values are configured as described in Table 2-1. The default combination is **nfcuser/nfcuser**.

Figure 2-1 CNS NetFlow Collection Engine User Interface Login Screen



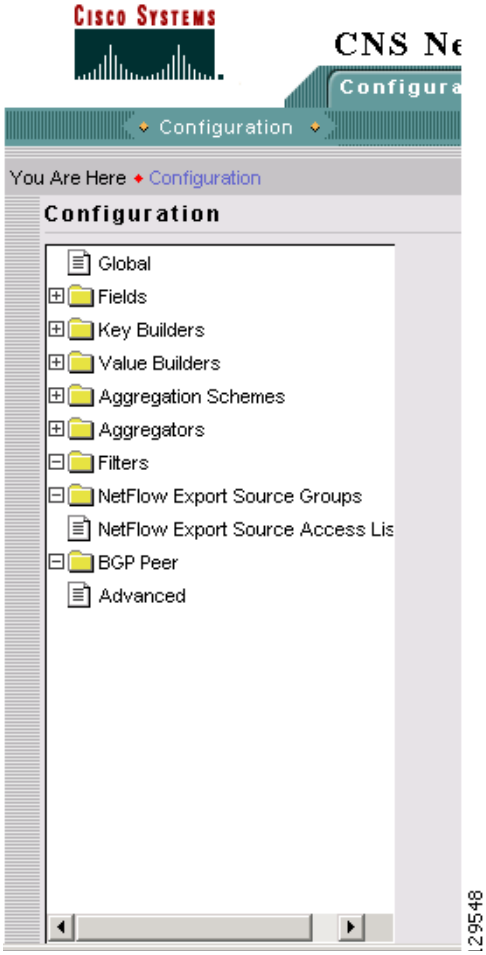
Navigation

Moving around the web-based UI is done at two levels. Across the top of all pages is the first level of web site navigation, as shown in Figure 2-2. From here you can switch between Configuration, Reports, and Status sections. The toolbar at the far right also includes links to Logout, Help, and About screens.

Figure 2-2 First Level Navigation of Web-based UI

Each section of User Interface has a navigation tree on the left-hand side, as shown in Figure 2-3. This second level of navigation lets you focus in on a specific aspect of collector configuration, reporting, or status.

Figure 2-3 Second Level Navigation of Web-based UI



Configuration

From the Configuration screen you can specify global parameters, define aggregators, and create filters.

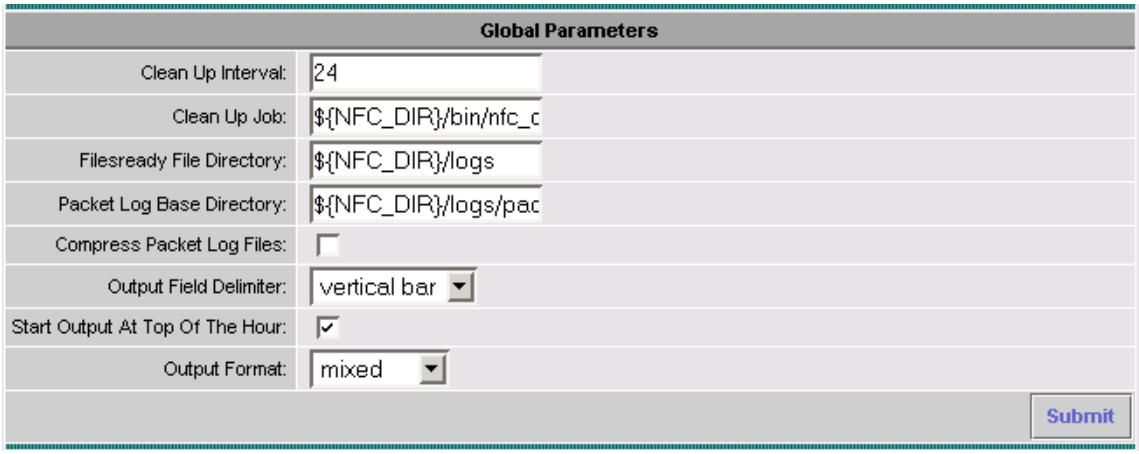
From the **Main** screen you receive upon logging in, click the **Configuration** tab and you will receive a screen as shown inFigure 2-4.

Figure 2-4 Configuration Screen

Global

The settings in Figure 2-5 affect how the CNS NetFlow Collection Engine works in general. They are not specific to any aggregator, aggregation-scheme, or filter. Make any changes necessary and click **Submit** to store them. Some settings do not take affect until the CNS NetFlow Collection Engine is restarted.

Figure 2-5 Global Parameters

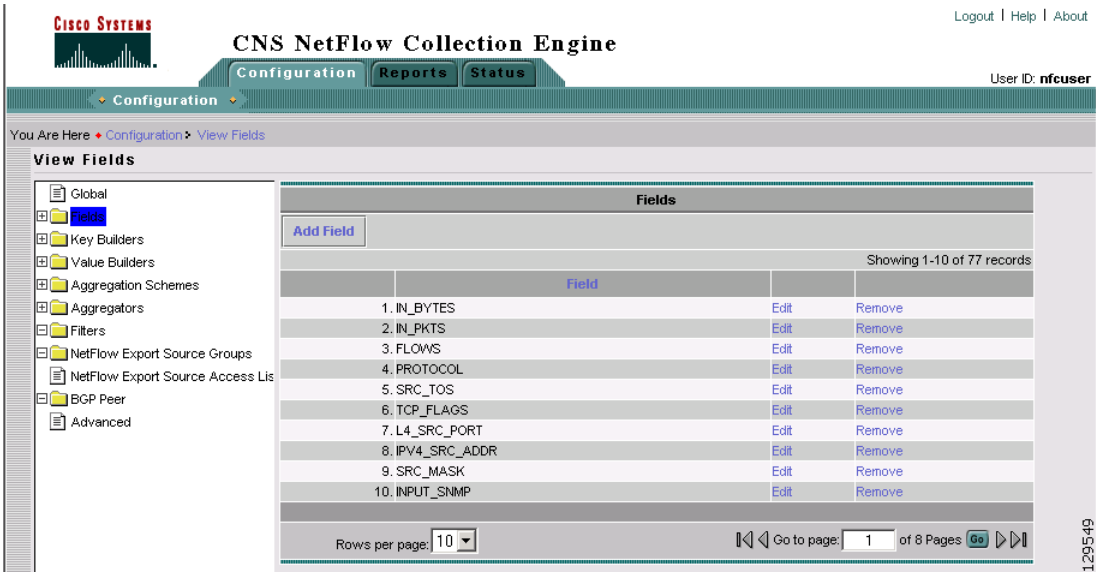


Fields

Fields represent individual items of data exported by a device in a NetFlow flow, and are the building blocks upon which the keys and values referenced by aggregation schemes are based.

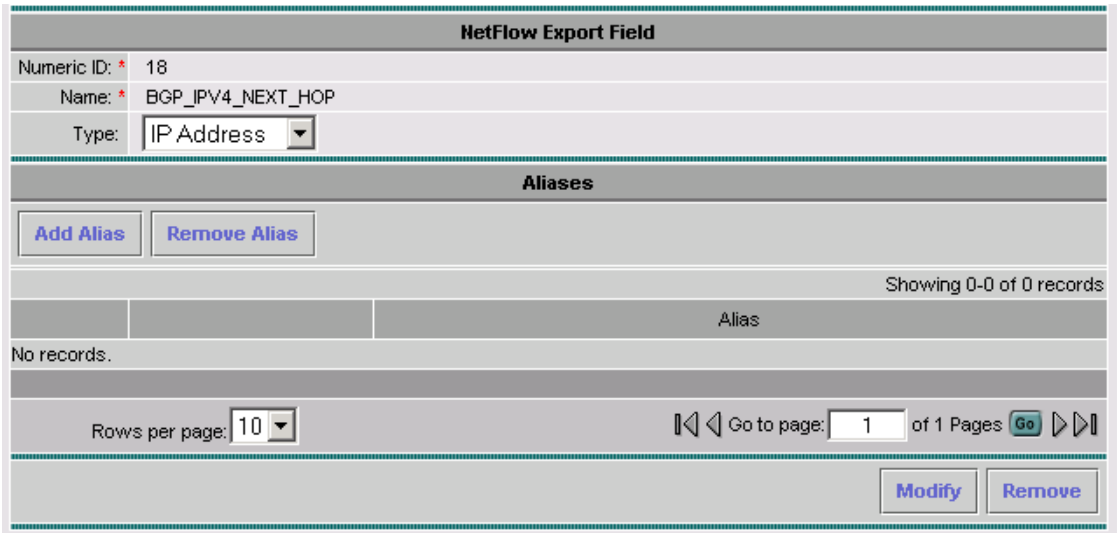
Clicking on the **Fields** folder of the navigation tree displays a table of currently defined fields as shown in Figure 2-6. Click the appropriate link to modify or remove a field. Click **Add Field** to bring up an empty form for defining a new field.

Figure 2-6 Add Field



The NetFlow Export Field screen, Figure 2-7, is displayed when adding or modifying a field. Fill in the form and click **Add** or **Modify** to complete the operation. From the Modify screen you may also remove the currently displayed field. Click **Add Alias** or **Remove Alias** to add or remove an alias (alternate name) for this field. Refer to the "Fields" section on page 4-4 for additional information about field definitions.

Figure 2-7 NetFlow Export Field



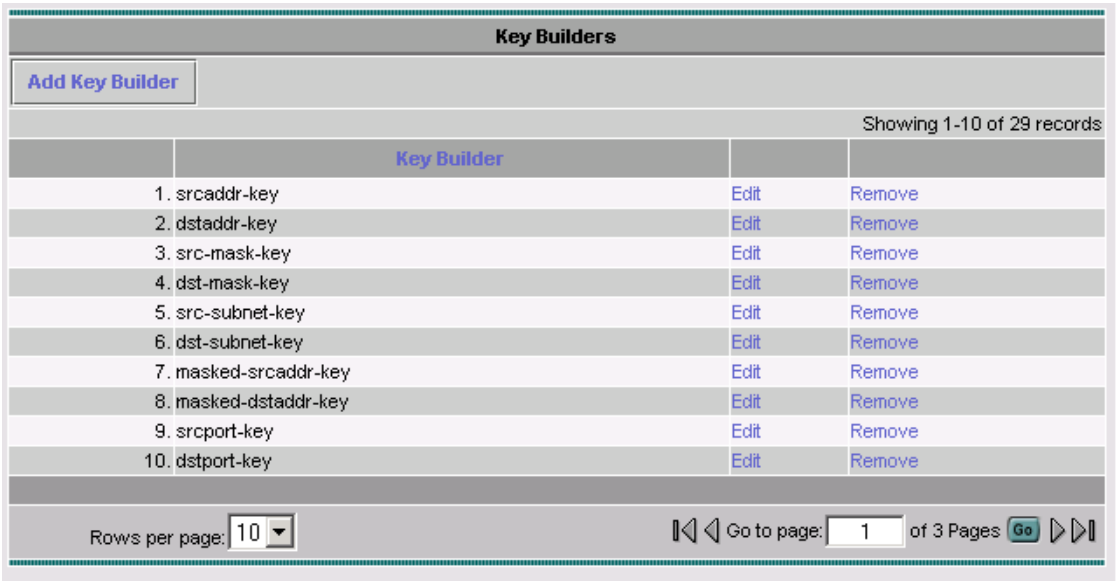
Key Builders

An aggregation scheme consists of *keys* and *values*. Within an aggregation period, each value within flows having the same set of keys is aggregated (typically summed) together with the corresponding values from earlier matching flows within an aggregation period.

Fields are not referenced directly by an aggregation scheme; instead, a *key builder* or *value builder* references a field, and one or more aggregation schemes references the builder.

Clicking on the **Key Builders** folder of the navigation tree displays a table of currently defined key builders as shown in Figure 2-8. Click the appropriate link to modify or remove a key builder. Click **Add Key Builder** to bring up an empty form for defining a new key builder.

Figure 2-8 Key Builders



Select the key builder in the navigation tree to modify it, or click the appropriate link in the table to modify or remove a key builder. Select **Add Key Builder** to bring up an empty form for defining a new key builder. All key builders have a unique ID and a type. The ID is displayed in the navigation tree and the key builder table. The attributes shown in the form depend on the type that is selected; different key builder types have different attributes. The following sections describe the attributes for each type of key builder.

Address Range Map

An **Address Range Map** key builder obtains an IP address from a flow and maps the value to a string. It has the following attributes.

Attribute	Description
Output name	Column name in output; defaults to the field ID if not specified.
Field	ID of the field to lookup from flows.
Allow null value	If set to *false* (default) and a flow does not contain field, an error is logged. I f set to *true*, the output value is empty and no error is logged.
Default label	Output value if no mapping result is found; otherwise if not specified the value itself is output.

Mapping information appears in the Address Ranges list. Each list item contains an IP address value or range and the label it maps to. Labels can appear more than once, but duplicate or overlapping values and ranges are not allowed. Click **Add range** to add a new value or range.

BGP Attribute

A **BGP Attribute** key builder looks up a BGP attribute from the CNS NetFlow Collection Engine BGP peer using an address from a flow. The complete AS path is a special case that uses both a source and a destination address from a flow. The BGP Attribute key builder has the following attributes.

Attribute	Description
Output name	Column name in output; defaults to the field ID if not specified.
Attribute type	Either a BGP attribute name from RFC1771; the integer type id; or the special case for the complete AS path.
Source address key	ID of a key builder that returns the source address for a complete AS path lookup, otherwise disabled.

Attribute	Description
Destination address key	ID of a key builder that returns the destination address for querying the attribute.
Post-aggregation	Determines whether lookups are performed for each flow or at the end of the aggregation period; this should always be selected, otherwise attributes are queried from the CNS NetFlow Collection Engine BGP peer as flows arrive resulting in a significant performance impact.

Bit Field

The **Bit Field** key builder obtains a subset of bits from a field in a flow. It has the following attributes.

Attribute	Description
Output name	Column name in output; defaults to the field ID if not specified.
Field	ID of the field in a flow from which to extract bits.
Least significant bit	Least significant bit of interest (starts at 0).
Number of bits	Number of bits of interest.
Format	*Decimal* or *hexadecimal*.
Allow null value	If *not* selected, an error is logged if a flow does not contain the indicated field.

Interface SNMP Name

The **Interface SNMP Name** key builder maps an interface index to an interface name obtained via SNMP. It has the following attributes.

Attribute	Description
Output name	Column name in output.
Field	ID of the field to obtain from a flow containing the interface index.
Allow null value	If *not* selected, an error is logged if a flow does not contain the indicated field.

Integer

An **Integer** key builder obtains an integer value from a flow. It has the following attributes.

Attribute	Description
Output name	Column name in output; defaults to the field ID if not specified.
Field	ID of the field in a flow.
Format	*Decimal* or *hexadecimal*.
Allow null value	If *not* selected, an error is logged if a flow does not contain the indicated field.

Integer Range Map

An **Integer Range Map** key builder obtains an integer from a flow and maps the value to a string. It has the following attributes.

Attribute	Description
Output name	Column name in output; defaults to the field ID if not specified.
Field	ID of the field in a flow.
Allow null value	If *not* selected, an error is logged if a flow does not contain the indicated field.
Default label	Mapping result if no match is found.

Mapping information appears in the Integer Ranges list. Each list item contains an integer value or range and the label it maps to. Labels can appear more than once, but duplicate or overlapping values and ranges are not allowed. Click on **Add Range** to add a new value or range.

Internet Address

An **Internet Address** key builder obtains an IP address from a flow. It has the following attributes.

Attribute	Description
Output name	Column name in output; defaults to the field ID if not specified.
Field	ID of the field in a flow.
Format	*Standard notation*, *hostname* (via a DNS lookup), or *integer*.
Allow null value	If *not* selected, an error is logged if a flow does not contain the indicated field.

Mask Field Internet Address

A **Mask Field Internet Address** key builder obtains both an IP address and mask value from a flow, then applies the mask to the address. It has the following attributes.

Attribute	Description
Output name	Column name in output.
Address field	ID of the address field to obtain from a flow.
Mask field	ID of the mask field to obtain from a flow.
Format	*Standard notation*, *hostname* (via a DNS lookup), or *integer*.
Allow null value	If *not* selected, an error is logged if a flow does not contain the indicated field.

Masked Internet Address

A **Mask Field Internet Address** key builder obtains an IP address from a flow then applies the specified mask to the address. It has the following attributes.

Attribute	Description
Output name	Column name in output.
Address field	ID of the address field to obtain from a flow.
Mask	Integer value of the mask to apply.
Format	*Standard notation*, *hostname* (via a DNS lookup), or *integer*.
Allow null value	If *not* selected, an error is logged if a flow does not contain the indicated field.

Multi-Field Map

The **Multi-Field Map** editor is applet-based and is different than the forms for other key builder types because of the hierarchical nature of a multi-field map. A tree on the left-hand side of the Multi-Field Map editor shows the elements of the map. A form on the right hand side of the Multi-Field Map editor shows the attributes for the selected item in the tree.

The top level of the tree contains the following attributes.

Attribute	Description
ID	ID that uniquely identifies this map.
Output name	Column name displayed in output for this key builder.
Default label	Default value shown in output if no match for the specified conditions is found.

Beneath the top level of the tree are one or more conditions. After selecting the top tree item, create a condition as follows:

1. Select the condition type (integer, IP address, or string).
2. Choose the key builder that will produce values for the condition.
3. Click **Add condition**.

A new condition will be added following all other conditions at that level and will be selected in the tree. The form displayed on the right side will display the new condition. In this form, select **Add case** one or more times to add cases for each value or range of interest. A new tree item for the case is added following all other cases under this condition's tree item; the new tree item is selected; and a form for the case is displayed on the right hand side.

A single case has one or more values and ranges and the label associated with a match for these values and ranges. The values and ranges for one case must be unique for all cases for this condition. To add a value or range to the case, select **Add value** or **Add range**. A new value or range is added to the case; a tree item for the value or range is added beneath the case's tree item; and a form is displayed on the right hand side for the new value or range.

Each case can also have one or more conditions nested beneath it that reference a different key builder. Therefore for a particular value, range, or set of values for one key, the value of a different key can further refine the result of the multi-field map. Conditions are added to a case as described above for adding conditions to the top level of the tree.

Selecting **Move** for a case or condition moves the tree item for the case or condition up. Once the item is at the top, it cycles back to the bottom. The order of cases has no impact on performance when evaluating a condition. However, because the conditions at one level in the tree are evaluated top-down in the order they appear, the order of conditions within one level can have an effect on performance. Therefore, if one condition is more likely than another, declare it first or move it before less likely conditions.

Any item in the tree including the items beneath it can be removed by selecting **Remove**. Pressing the back button on the browser also causes any changes to be discarded. Remove items with care because no cut, paste, or undo capability is provided. Changes are not committed until you select **Update map** or **Remove map**.

The symbol [**!**]at the beginning of any item in the tree indicates that the configuration specified at that level of the tree is incomplete and must be updated before the multi-field map can be added or updated.

Option Data

An **Option Data** key builder obtains one or more key values from a flow and performs a lookup using this result from an option data cache. The result of the mapping is the corresponding value from option data that was specified in the option data cache entry definition. The **Option Data** key builder has the following attributes.

Attribute	Description
Output name	Column name in output.
Option data map entry	ID of an option-data-map-entry element declared in option-data-map in XML configuration.
Keys	ID of one or more key builders to produce values corresponding with the keys in the specified option-data-map-entry.

String

A **String** key builder obtains a UTF-8 string value from a flow. It has the following attributes.

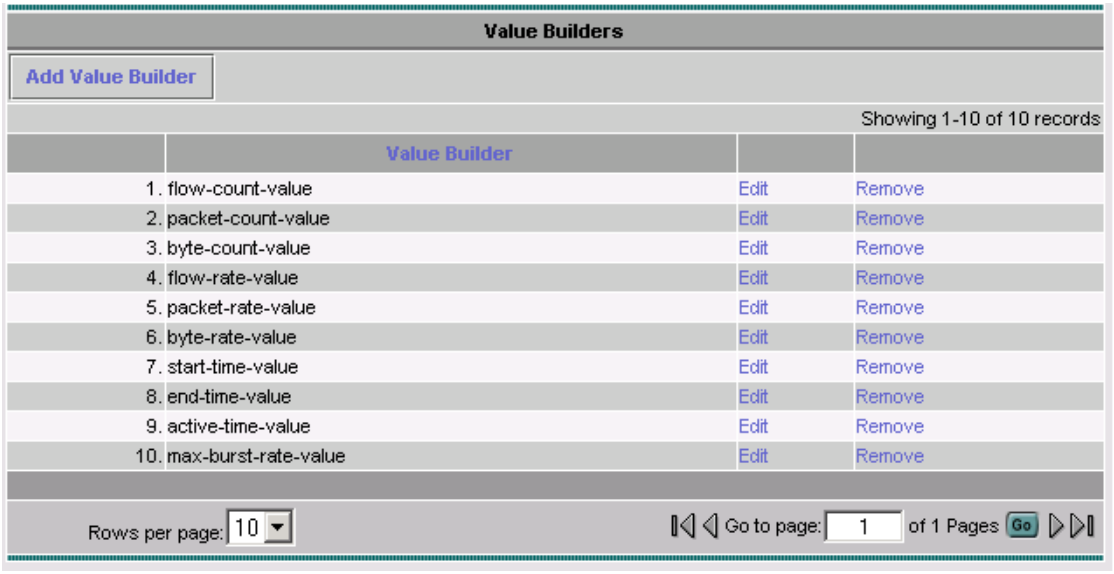
Attribute	Description
Output name	Column name in output.
Field	ID of the field to obtain from a flow.
Allow null value	If *not* selected, an error is logged if a flow does not contain the indicated field

Value Builders

A value builder is associated with one or more fields in flow data and produces a non-key value in an aggregation record. A value builder can be referenced by an Aggregation Scheme and corresponds with one column in an CNS NetFlow Collection Engine output file.

Clicking on the **Value Builders** folder of the navigation tree displays a table of all existing value builders, as shown in Figure 2-9. Click on the appropriate link to modify or remove a value builder.

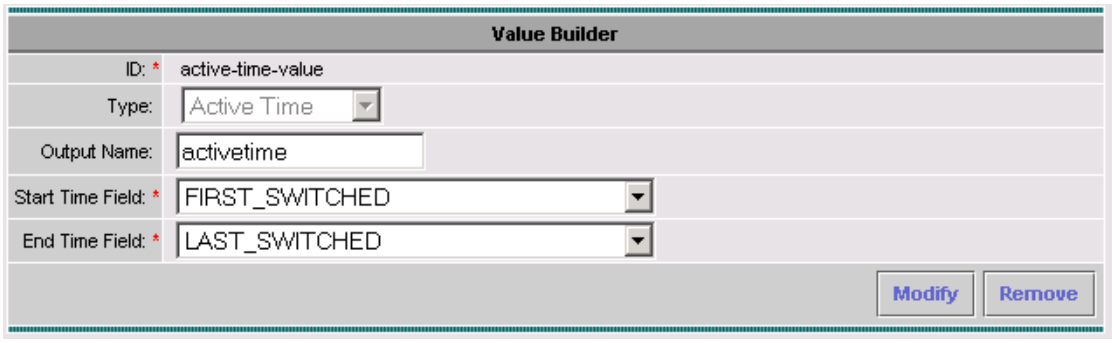
Figure 2-9 Value Builders



Click on **Add Value Builder** to bring up an empty form for defining a new value builder. A value builder is created by specifying its type, associating it with a field (sometimes two or more fields such as for the Active Time type as shown in Figure 2-10), and specifying attributes specific to the selected type. Different forms are displayed depending on which value builder type is selected.

When **Add Value Builder** or **Edit** is selected, a form for editing the value builder definition is displayed. All value builders have an ID and Type. The ID must be unique for all value builders; the Type determines the algorithm used to create the value. The remaining attributes that are shown in the Value Builder form are determined by which type is selected.

Figure 2-10 Adding a Value Builder



Refer to the "Keys and Values" section on page 4-5 for additional information about value builder definitions.

Active Time

The **Active Time** value builder obtains a start time and an end time from fields in a flow and calculates the difference. It has the following attributes.

Attribute	Description
Name	Column name in output.
Start time field	ID of the start time field to obtain from a flow.
End time field	ID of the end time field to obtain from a flow.
Usage	Always leave set as **Count**.

End Time

The **End Time** value builder obtains an end time from a field in a flow. It has the following attributes.

Attribute	Description
Name	Column name in output.
End time field	ID of the end time field to obtain from a flow.

Flow Count

The **Flow Count** value builder increments a count for each flow. It has the following attributes.

Attribute	Description
Name	Column name in output.
Usage	Always leave set as **Count**.

Max Burst Rate

The **Max Burst Rate** value builder determines a burst rate from the byte count, start time, and end time in a flow and outputs the highest value found for all flows in an aggregation period. It has the following attributes.

Attribute	Description
Name	Column name in output.
Start time field	ID of the start time field to obtain from a flow.
End time field	ID of the end time field to obtain from a flow.
Byte count field	ID of the byte count field to obtain from a flow.
Usage	Always leave set as **Maximum**.

Rate

The **Rate** value builder determines a rate by dividing the result of another value by the amount of time in the aggregation period. It has the following attributes.

Attribute	Description
Name	Column name in output.
Quantity value	ID of another value builder used to determine the quantity.
Units	Scales the result to seconds or minutes.

Start Time

The **Start Time** value builder obtains a start time from a field in a flow. It has the following attributes...

Attribute	Description
Name	Column name in output.
Start time field	ID of the start time field to obtain from a flow.

Sum

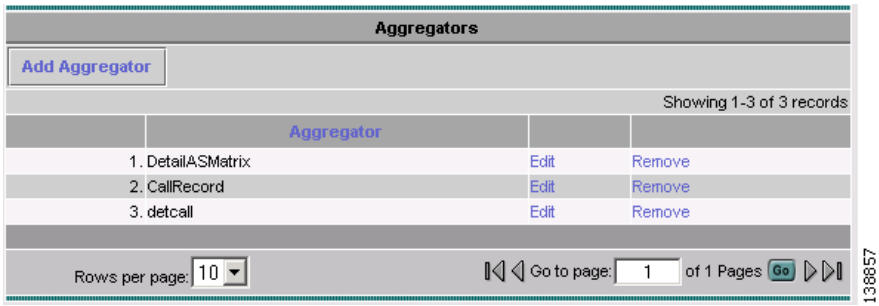
The **Sum** value builder obtains an integer value from a field in a flow and adds it to a count. It has the following attributes...

Attribute	Description
Name	Column name in output.
Field	ID of the integer field to obtain from a flow.
Usage	Always leave set as **Count**.

Aggregators

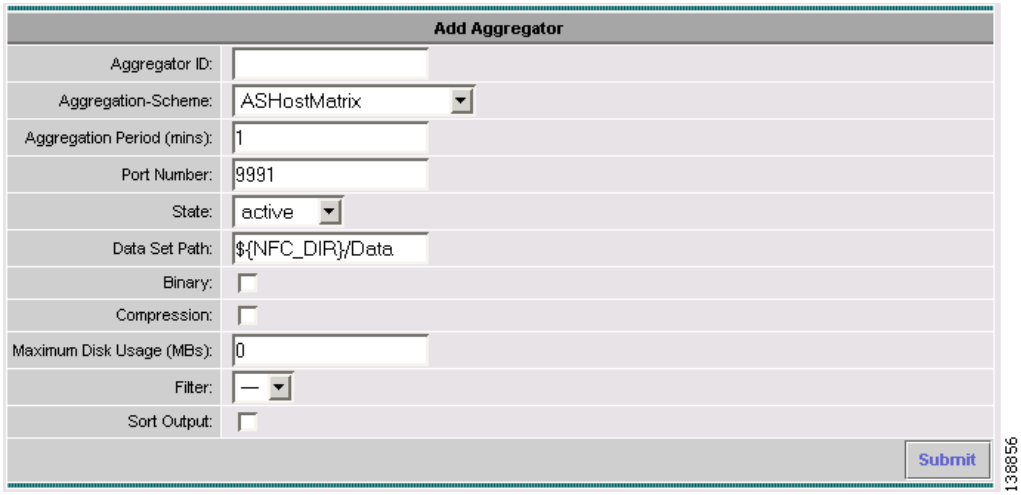
Aggregators define how the CNS NetFlow Collection Engine receives NetFlow data, aggregates it, and generates output files. As shown in Figure 2-11, clicking on the Aggregators folder of the navigation tree displays a table of all existing aggregators.

Figure 2-11 Aggregators



Click on **Add Aggregator** to bring up the Add Aggregator screen to define a new aggregator, as shown in Figure 2-12.

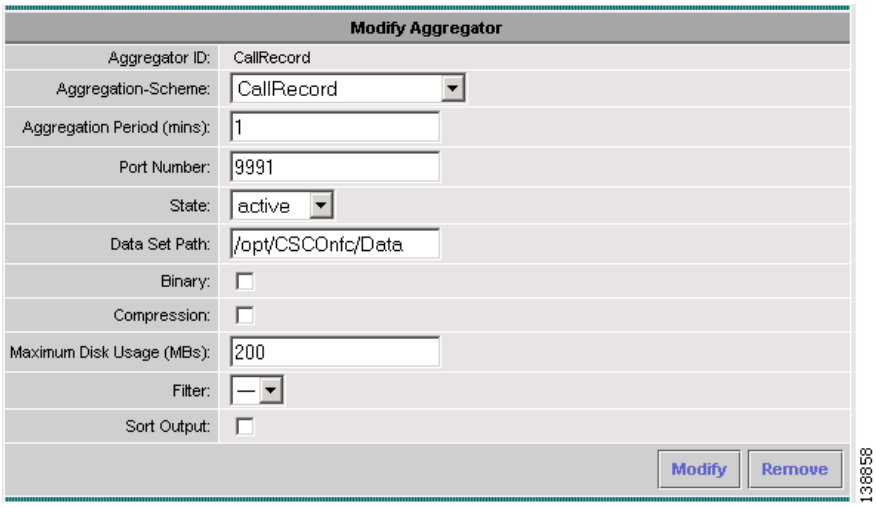
Figure 2-12 Add Aggregators



Fill in the fields and click **Submit** to complete the operation.

To modify an existing aggregator, click **Edit** for the aggregator which you wish to modify from the list of aggregators displayed in the Aggregator screen. The **Modify Aggregator** screen displays, as shown in Figure 2-13.

Figure 2-13 Modify Aggregators



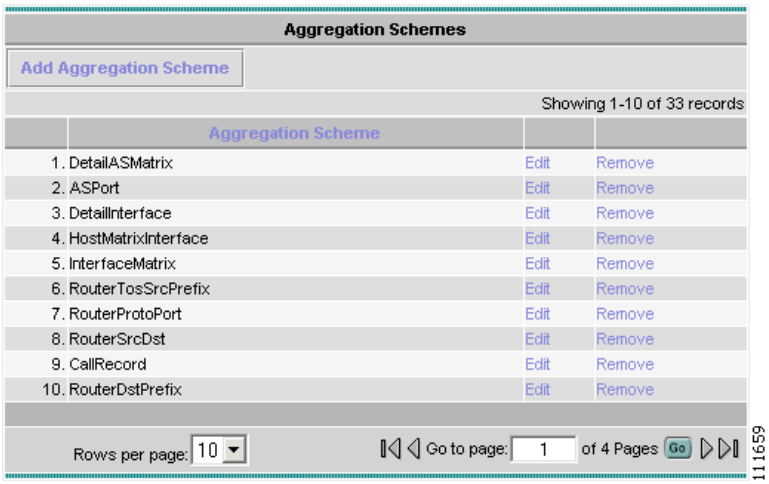
Fill in the fields and click **Modify** to complete the operation. From the Modify Aggregator screen you can also remove the currently displayed aggregator.

Note When a key or value builder, filter, or aggregation scheme is modified through the web-based user interface, collector configuration is updated immediately. However, for the update to have an affect on aggregation and output, the aggregator must be modified or the collector must be restarted.

Aggregation Schemes

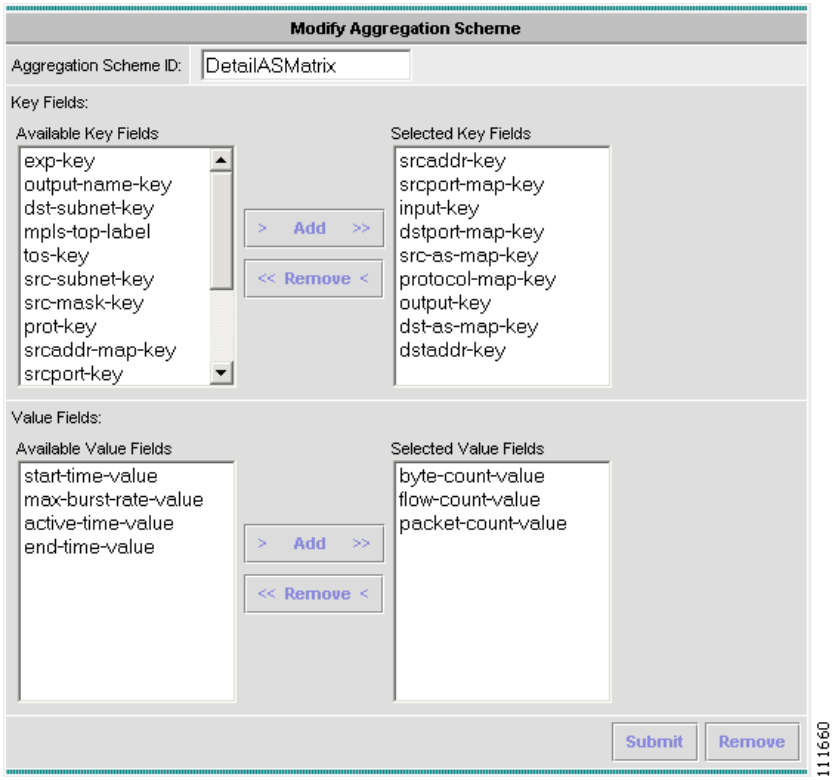
Aggregation schemes define the set of keys and values used for aggregation and that appear in the CNS NetFlow Collection Engine output files. Clicking on the **Aggregation Schemes** folder of the navigation tree displays a table of all existing aggregation schemes, as shown in Figure 2-14. Click on the appropriate link to modify or remove an aggregation scheme. Click on **Add Aggregation Scheme** to bring up an empty form for defining a new aggregation scheme.

Figure 2-14 Aggregation Schemes



The **Add Aggregation Scheme** and **Modify Aggregation Scheme** in screens, as shown in Figure 2-15, are identical with the exception that you can not change the Aggregation Scheme ID on the Modify Aggregation Scheme screen. Use this form to select key and value fields and click **Add** or **Modify** respectively to complete the operation. From the **Modify Aggregation Scheme** screen you can also remove the currently displayed aggregation scheme.

Figure 2-15 Modify Aggregation Scheme

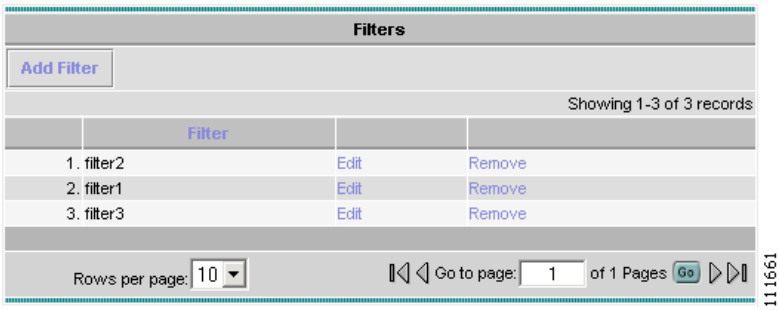


Note Removing an aggregation scheme that is in use by an aggregator can succeed but cause an invalid reference after the collector is restarted.

Filters

Filters provide a way to limit the amount and content of data that an aggregator processes. Clicking on the **Filters** folder of the navigation tree displays a table of all existing filters, as shown in Figure 2-16. Click on the appropriate link to modify or remove a filter. Click on **Add Filter** to bring up an empty form for defining a new filter.

Figure 2-16 Filters



When adding and editing filters the screens are identical with the exception that you cannot change the **Filter ID** when modifying a filter. Use this form to add, remove and order filter conditions.

The Filter editor is applet-based. A tree on the left hand side of the filter editor shows the elements of the filter. A form on the right hand side of the filter editor contains the attributes for the currently selected item in the tree.

The top item of the tree contains a unique identifier for the filter. Directly beneath the top of the tree is one filter condition or filter expression. Add the top-level filter condition or expression by selecting Add condition or Add expression when the top item is selected.

A filter condition performs an equality check on the output value of a key builder that is invoked for each flow. The type of a filter condition is either an integer condition, address condition, string condition, or nde-source condition. Depending on which condition type you select, only the key builders that produce that type of value can be selected. The nde-source condition checks the address of the device from which the flow originated.

When creating a filter condition, specify:

- Whether the equality check is **equals** or **not-equals**
- Which key builder creates the value to be checked

In addition, an address condition accepts an optional integer mask value that is applied to the address before the equality check is performed. If the mask field is left blank, no mask is applied.

Directly beneath the filter condition is one or more value or range items. These determine the set of target values to which the equality check is applied. Add a value or range to the filter condition by selecting **Add value** or **Add range**. For an integer condition, only integer values and ranges can be entered; only IP address values can be entered for address filter conditions. An nde-source condition accepts only IP address values. Note that ranges cannot be entered for string filter conditions, only single values.

Boolean logic is applied to two or more filter conditions using a filter expression. A filter expression can also appear within an expression in place of a filter condition.

To create a filter expression, specify the logical operator **and**, **or**, **nand** (not-and), or **nor** (not-or) and select **Add expression**. An expression must contain at least two other conditions or expressions.

The conditions and expressions within an expression are evaluated in top-down order. Evaluation performance for an expression can be optimized by placing conditions and expressions which are more likely to occur to the top. Select an item then select **Move** to move the item up until it reaches the top; selecting **Move** again cycles the item to the bottom.

Any item in the tree including the items beneath it can be removed by selecting **Remove**. Pressing the back button on the browser also causes any changes to be discarded.

Note Remove items with care since no cut, paste, or undo capability is provided. Changes are not committed until you select **Update filter** or **Remove filter**.

The symbol [**!**] at the beginning of any item in the tree indicates that the configuration specified at that level of the tree is incomplete and must be updated before the filter can be added or updated.

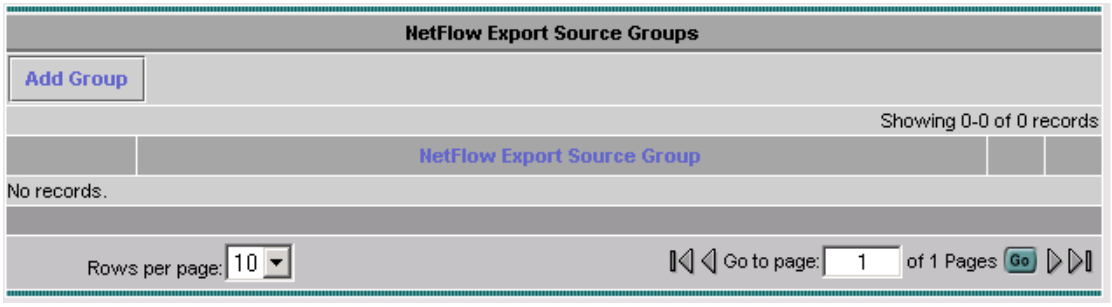
NetFlow Export Source Groups

By default, flows are aggregated with other flows from the source address of the originating device. However, if multiple source addresses appear in one export Source Group, flows from these multiple sources are aggregated together.

Note The collector must be restarted for configuration changes to an existing source group to take effect.

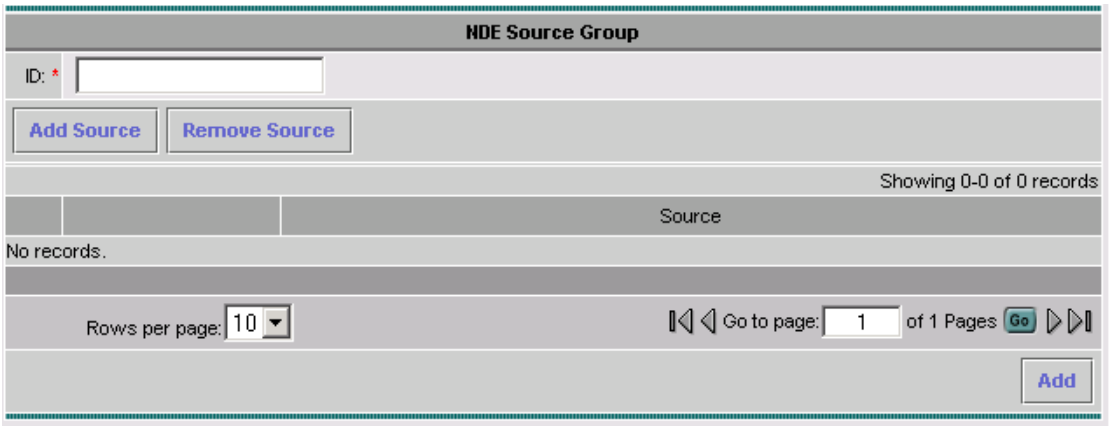
Click on the **NetFlow Export Source Groups** folder of the navigation tree to display a table of currently defined source groups, as shown in Figure 2-17. Click on the appropriate link to modify or remove a group. Click on **Add Group** to bring up an empty form for defining a new source group.

Figure 2-17 NetFlow Export Source Groups



The **NDE Source Group** screen, as shown inFigure 2-18, is shown when adding or modifying a source group. Fill in the form and click **Add** or **Modify** to complete the operation. Select **Add Source** to add an IP address to the group. From the **Modify** screen you may also remove the currently displayed source group. Refer to the "Creating Source Groups" section on page 4-24 for additional information about source groups.

Figure 2-18 NDE Source Group



NetFlow Export Source Access List

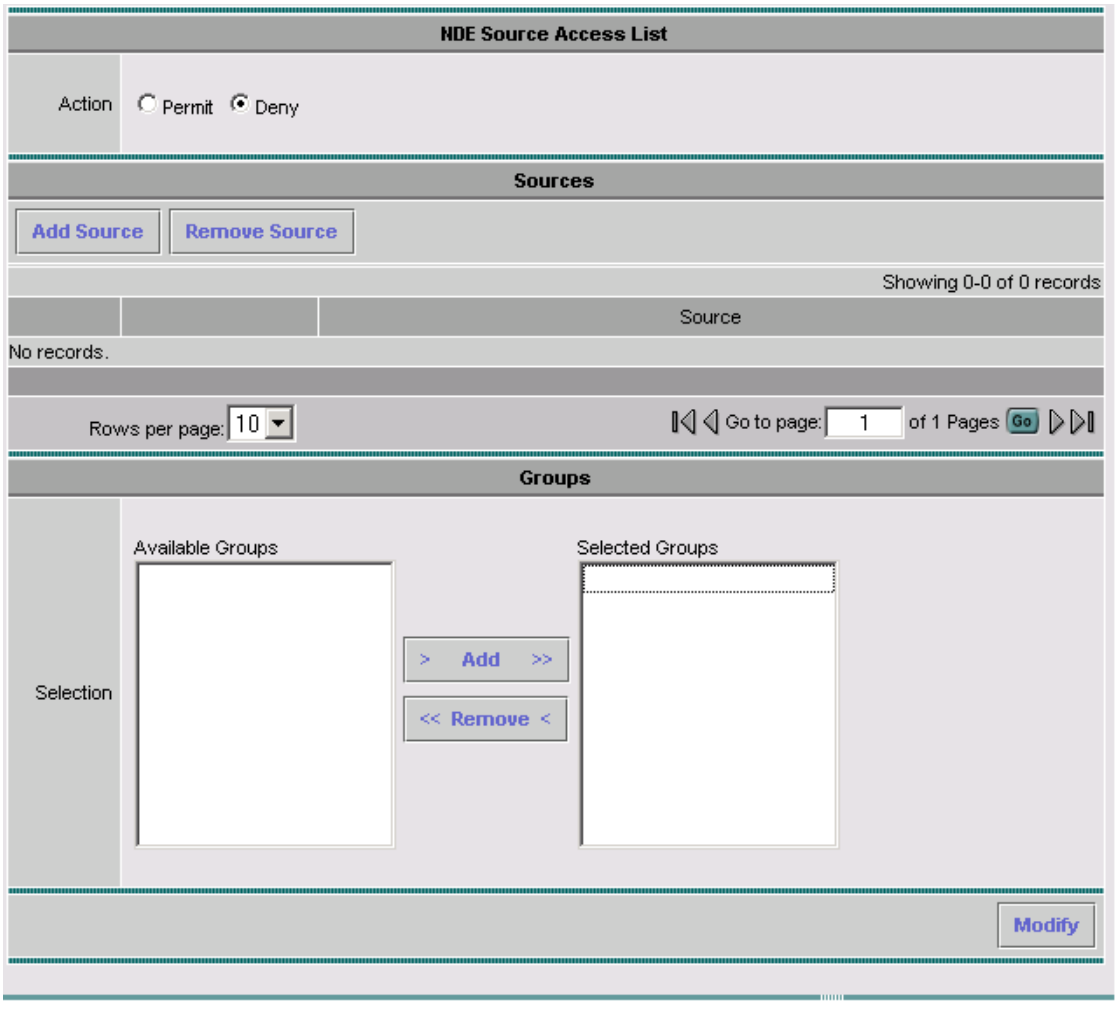
By default, CNS NetFlow Collection Engine collects from any device that sends NetFlow data to it. However, by specifying a NetFlow Export Source Access List, you can configure CNS NetFlow Collection Engine to reject data from certain devices or to accept data only from certain devices.

Note The collector must be restarted for configuration changes to the source access list to take effect.

Click on the **NetFlow Export Source Access List** folder of the navigation tree to display the current access list, as shown in Figure 2-19. If Action is **Permit**, NetFlow data is permitted only from the selected devices and groups; if Action is **Deny**, NetFlow data is rejected from the selected devices and groups.

Click on the appropriate link to add or remove a source device or group. Note that groups are obtained from the NetFlow Export Source Groups page. Refer to the "Creating Access Lists" section on page 4-24 for additional information about configuring source access lists.

Figure 2-19 NDE Source Access List

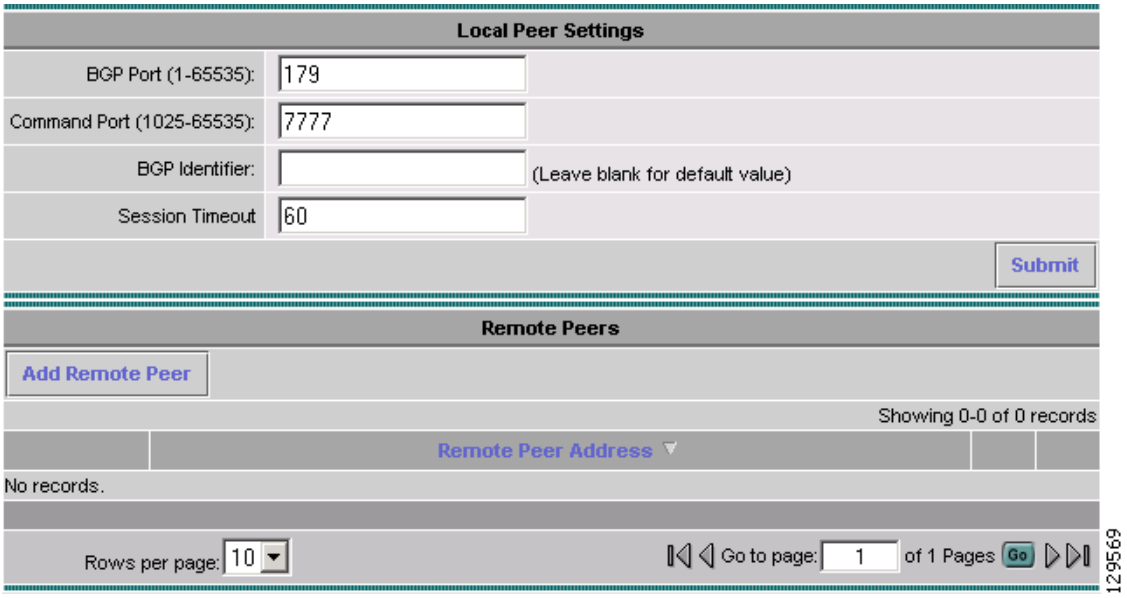


BGP Peer

Click the **BGP Peer** folder of the navigation tree to display the configuration for the CNS NetFlow Collection Engine BGP peer, as shown in Figure 2-20. Click on **Add Remote Peer** to specify a new BGP peer. If the BGP Identifier field is left blank, the BGP identifier of the CNS NetFlow Collection Engine BGP peer defaults to the integer value of this host's IP address.

Note The BGP Peer must be stopped and restarted for configuration updates to take effect. Refer to the "BGP Peer" section on page 5-8 for additional information about BGP Peer configuration.

Figure 2-20 Peer Settings



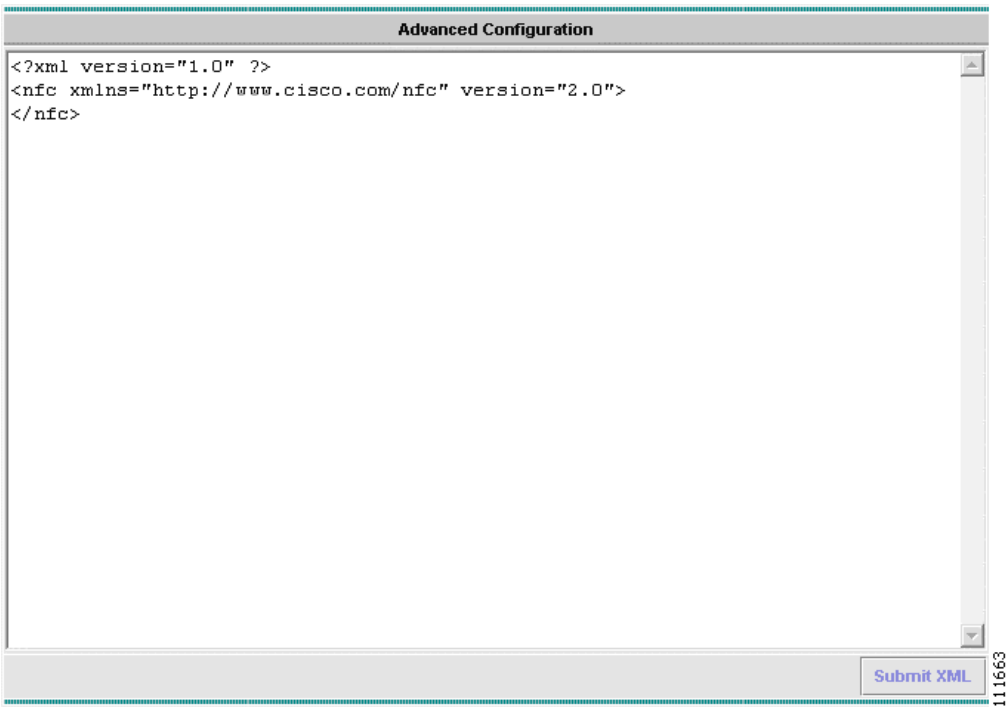
Advanced

The Advanced screen lets you send any XML request to the collector. Clicking on the **Advanced** node in the navigation tree brings up a form with a template for an XML request. Add the content of the XML request inside the **<nfc>** tag. See the "Supported XML Requests" section on page E-3 for a description of valid XML requests.

Some CNS NetFlow Collection Engine configuration is more complex than the web-based UII supports. For instance, complex filter expressions are not supported by the Modify Filter screen. In such cases, you will be directed to the **Advanced** screen and the XML for the selected component (aggregator, aggregation-scheme, filter) will appear in the text area. Changes can then be made and submitted by clicking **Submit XML**.

XML responses from the collector are displayed in Figure 2-21 in the text area after submitting a request.

Figure 2-21 Advanced Configuration

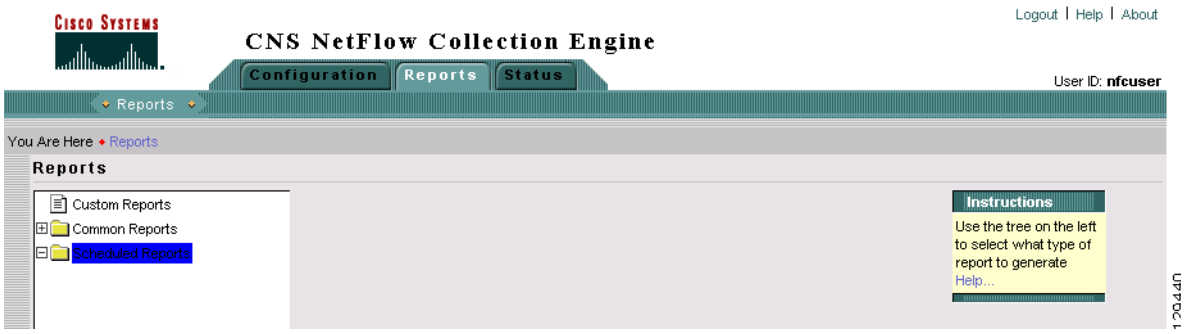


Reports

CNS NetFlow Collection Engine reports are in effect a summary of CNS NetFlow Collection Engine aggregated output. NetFlow data is first aggregated into CNS NetFlow Collection Engine output files by the collector, and then the data in those files is further aggregated to generate a report. Reports are categorized into three types: custom, common, and scheduled.

From the **Main** screen you receive upon logging in, click the **Reports** tab and you will receive a screen as shown inFigure 2-22.

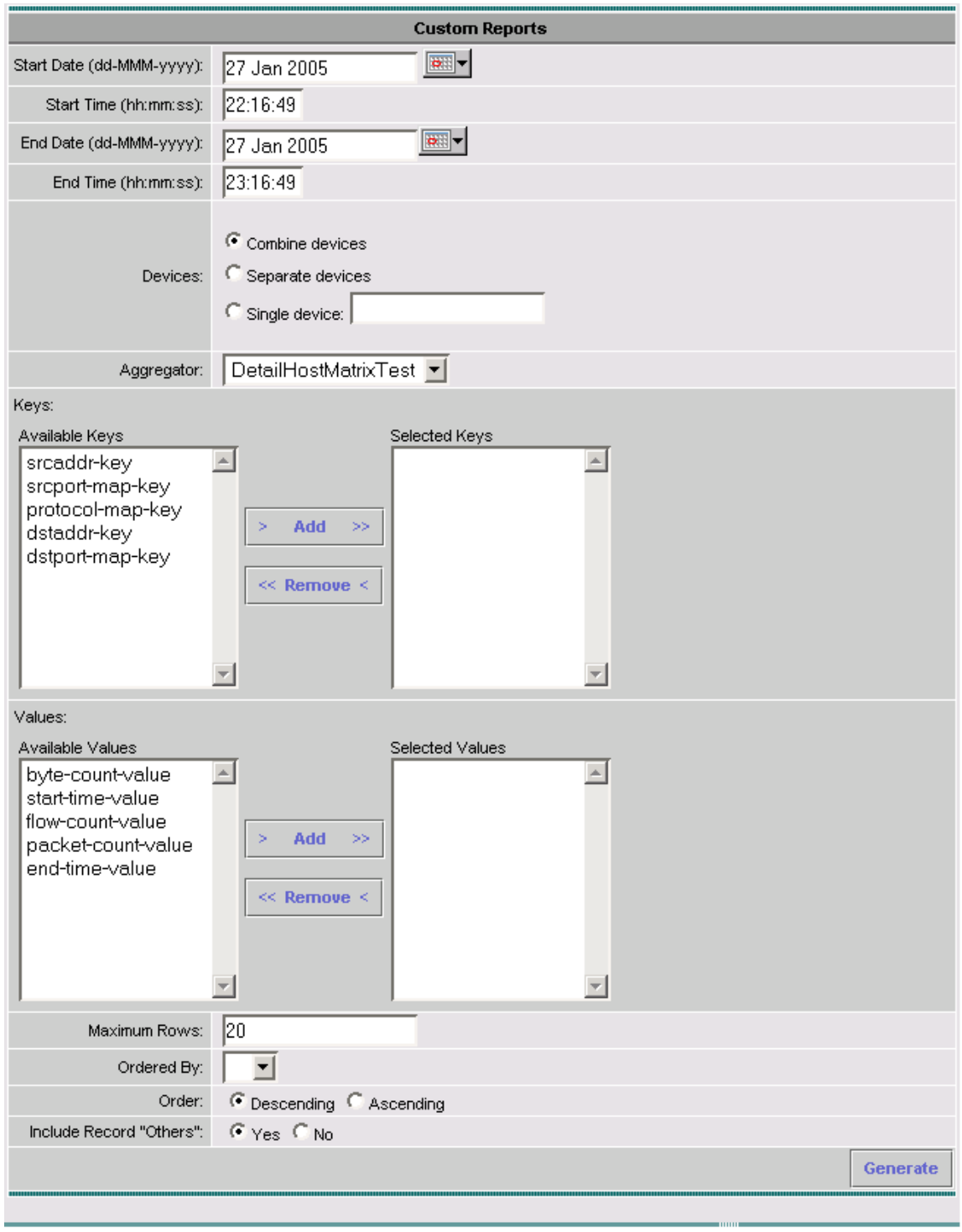
Figure 2-22 Reports Screen



Custom Reports

Custom reports are generated dynamically from the set of CNS NetFlow Collection Engine output files accessible from the collector machine. Use the **Custom Reports** form, as shown in Figure 2-23, to specify which data is included in the report and how it is aggregated.

Figure 2-23 Custom Reports



The fields of the Custom Reports form are described in Table 2-2.

Table 2-2 Custom Reports Fields

Field	Value	Description
Start Date	A date string in the format of dd-MMM-yyyy where dd is the day of the month, MMM is the abbreviated name of the month, and yyyy is the four digit year. For example, 01-Jan-2004 for January 1st, 2004.	The data for the report will come from CNS NetFlow Collection Engine output files that were generated on or after this date.
Start Time	A time string in the format of hh:mm:ss where hh is the hour of the day in 24 hour notation, mm is the minute of the hour, and ss is the seconds of the minute. For example, 13:05:00 for 1:05PM and 0 seconds.	The data for the report will come from CNS NetFlow Collection Engine output files that were generated at or after this time.
End Date	A date string in the format of dd-MMM-yyyy where dd is the day of the month, MMM is the abbreviated name of the month, and yyyy is the four digit year. For example, 01-Jan-2004 for January 1st, 2004.	The data for the report will come from CNS NetFlow Collection Engine output files that were generated on or before this date.
End Time	A date string in the format of dd-MMM-yyyy where dd is the day of the month, MMM is the abbreviated name of the month, and yyyy is the four digit year. For example, 01-Jan-2004 for January 1st, 2004.	The data for the report will come from CNS NetFlow Collection Engine output files that were generated at or before this time.
Devices	**Combine devices**, **Separate devices**, or **Single device**. For **Single device** the value should be the IP address of the device.	**Combine devices** specifies that the report will aggregate data from different exporting devices into records based solely on the specified keys (See below). Each row of the report will contain a * for the value of the Device column. **Separate devices** specifies that the report will treat the exporting device as an additional key for aggregation. As a result, data from different devices will not be aggregated together and the exporting device that generated the report data will be the value of the Device column for each row of the report. **Single device** allows you to filter report data to that which came from a single exporting device. The IP address of the exporting device will be the value of the Device column for each row of the report.

Table 2-2 Custom Reports Fields (continued)

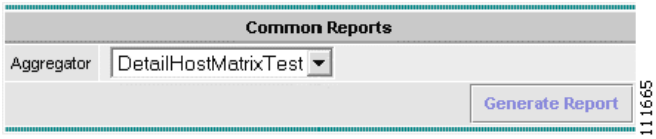
Field	Value	Description
Aggregator	One of the defined aggregators	The report data will come from the CNS NetFlow Collection Engine output files of this single aggregator.
Keys	The set of keys that are defined in the aggregation scheme used by the selected aggregator, or a subset of these keys.	Report data will be aggregated for each unique combination of keys selected for the report. Using a subset of keys reduces the system memory required to generate the report.
Values	The set of values that are defined in the aggregation scheme used by the selected aggregator, or a subset of these values.	Value columns of the report are aggregated for each unique combination of keys selected for the report. Using a subset of values reduces the system memory required to generate the report.
Maximum Rows	A positive integer, *N*, no greater than 2147483647. Default value is 20.	The maximum number of rows the report should contain for each exporting device. The total number of unique records in all the CNS NetFlow Collection Engine data files being reported can be much greater than the number of the records one may want to present in a report. Use this field to limit the number of records contained in the report. You can sort all aggregated unique records in descending (or ascending) order, according to a user-specified value field, and present the first *N* records in the report. The remaining records can be optionally aggregated into one record with key value of **Others**.
Ordered By	One of the selected values. The default value is the first selected value.	The sorting of all aggregated unique records will be performed based on this field.
Order	**Descending** or **Ascending**. The default value is **Descending**.	The order in which the aggregated records are sorted. When the order is set to **Descending**, the first *N* records presented in the report will be the first *N* records displayed; in contrast, if **Ascending** is chosen, the first *N* records in the report will be the last *N* records. displayed
Include Record **Others**	**Yes** or **No**. The default value is **Yes**.	Specifies whether to include the record with key value of **Others**. If set to **Yes**, the **Others** record will be calculated and appear in the report. If set to **No**, the report will not contain the **Others** record.

Note Rate values are not supported in custom reports in this release.

Common Reports

Common reports are also dynamically generated reports that simplify the creation process by predefining most of the report criteria. First select a common report from the navigation tree. From the **Common Reports** screen, as shown in Figure 2-24, select an aggregator from which the report data will come. Finally, click **Generate Report** to create the report.

Figure 2-24 Common Reports



There are four Common Reports to choose from:

- Today's Top Talkers
- Yesterday's Top Talkers
- Today's Protocol Distribution
- Yesterday's Protocol Distribution

The Top Talkers reports use the source address for the sole key column and the byte count for the sole value column. The Protocol Distribution reports use protocol for the sole key column and the byte count for the sole value column.

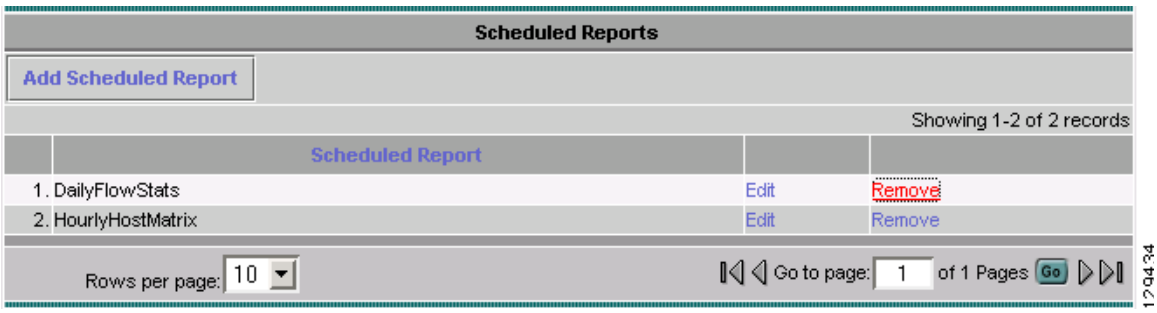
Scheduled Reports

Scheduled reports are generated by the Report Generator on a regular basis. Beginning with CNS NetFlow Collection Engine 5.0.2, the Report Generator supports running multiple types of reports simultaneously. You can configure the scheduled reports using the web-based UI.

Configuring Scheduled Reports

Clicking on the **Scheduled Reports** folder in the navigation tree displays a table of all existing types of scheduled reports, as shown in Figure 2-25.

Figure 2-25 Scheduled Reports



Clicking **Add Scheduled Report** brings up the **Add Scheduled Report** screen to add a new scheduled report. Clicking **Edit** in any row in the list of scheduled reports displays the **Modify Scheduled Report** screen to modify the selected scheduled report. Clicking **Remove** in any row deletes the selected schedule report. The **Add Scheduled Report** and **Modify Scheduled Report** screens, as shown in Figure 2-26, are identical with the exception that you can not change the **Report ID** on the **Modify Scheduled Report** screen. Fill in the fields and click **Submit** or **Modify** button to complete the operation.

Note Configuration updates for scheduled reports via the UI will not take effect until the Report Generator is restarted.

Figure 2-26 Add Scheduled Report

Scheduled Report screens share many commonalities with the Custom Report screen, but there are a few differences:

- There is no Start Date, Start Time, End Date and End Time fields on Scheduled Report screens, because these values are pre-determined. For daily reports, the start time is at the turn of the day and end time the turn of the next day; for hourly reports, similarly, the start time is the turn of the hour and end time the turn of the next hour.
- There are four additional fields. See Table 2-3 for descriptions.

Table 2-3 Scheduled Report Fields

Field	Value	Description
Scheduled Report ID	String containing alphanumeric characters plus a hyphen (-) and underscore (_).	The ID to identify this type of report.
Report Frequency	**Daily** or **Hourly**. The default value is **Daily**.	The frequency at which this type of report is run.
Days To Keep	A positive integer no greater than 32767. The default value is 7.	The number of days the generated reports of this type will be kept on the server. Reports of this type past this date will be purged automatically.
Output Path	Place-name of an existing directory. The default value is **/opt/CSCOnfc/Reports**.	Specifies where reports of this type will be stored. All reports of this type will be written to the subdirectory (named with the report ID) under the output path. For example, if you use the default output path **/opt/CSCOnfc/Reports** and the report ID is **foo**, all reports of type **foo** will be stored in **/opt/CSCOnfc/Reports/foo**.

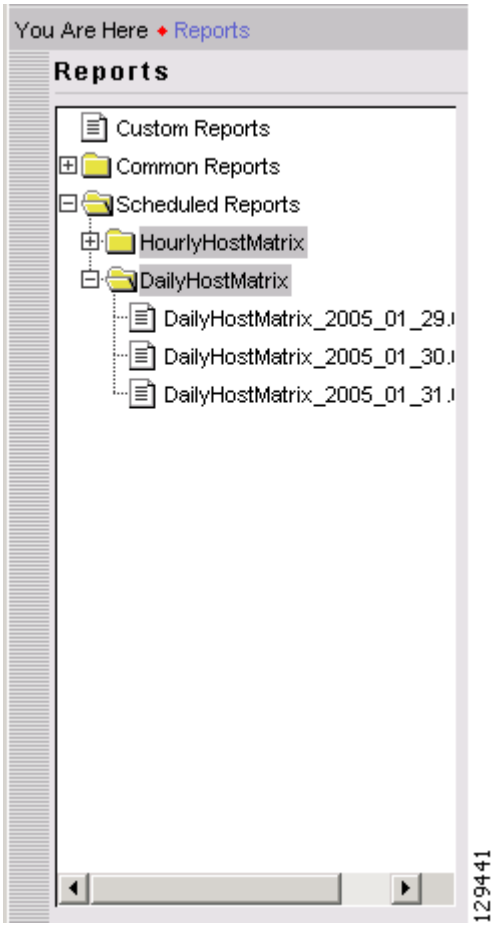
Note Rate values are not supported in scheduled reports in this release.

Displaying Scheduled Reports

You can use the web-based UI to view scheduled reports. The IDs of all types of defined reports display in the Reports navigation tree as subfolders of the Scheduled Reports folder, as shown in Figure 2-27. Reports generated by the Report Generate and placed in user-specified directories display as children (or leaf nodes) in the subfolders of the corresponding report type. Clicking on a report node brings up a window with that report displayed. Reports stored in the CNS NetFlow Collection Engine report XML format are formatted into tabular form. Reports stored in other formats are loaded as is and the presentation is left to the browser.

Note Scheduled reports do not support the advanced features, such as (Filter and Drill Down) of Custom and Common reports.

Figure 2-27 Scheduled Reports Folder



Reporting Features

CNS NetFlow Collection Engine reports enable you to sort, graph, export, filter, and drill down on report data from the Report screen, as shown in Figure 2-28.

Figure 2-28 Report

Sorting and Graphing

Each column of a report supports ascending and descending sorting. Click on the column name to sort the table on that column. Value columns support creating a bar or pie graph of the values in that column Click on the bar graph icon to generate a bar graph of that column's values, as shown in Figure 2-29. Click on the pie graph icon to generate a pie graph of that column's values, as shown in Figure 2-30.

Figure 2-29 Sample Bar Graph

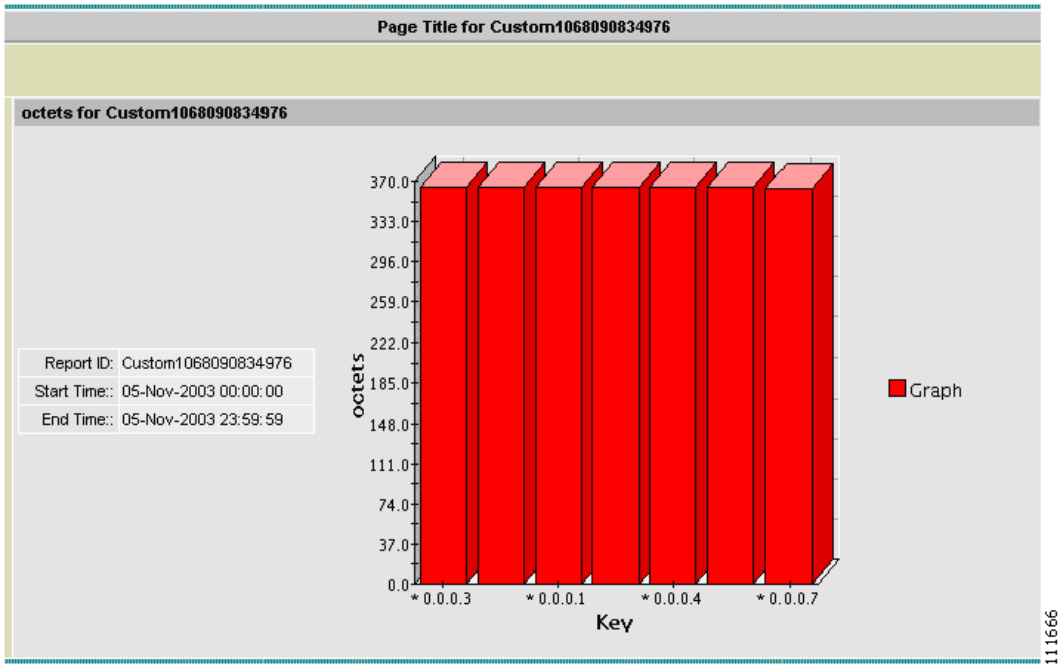
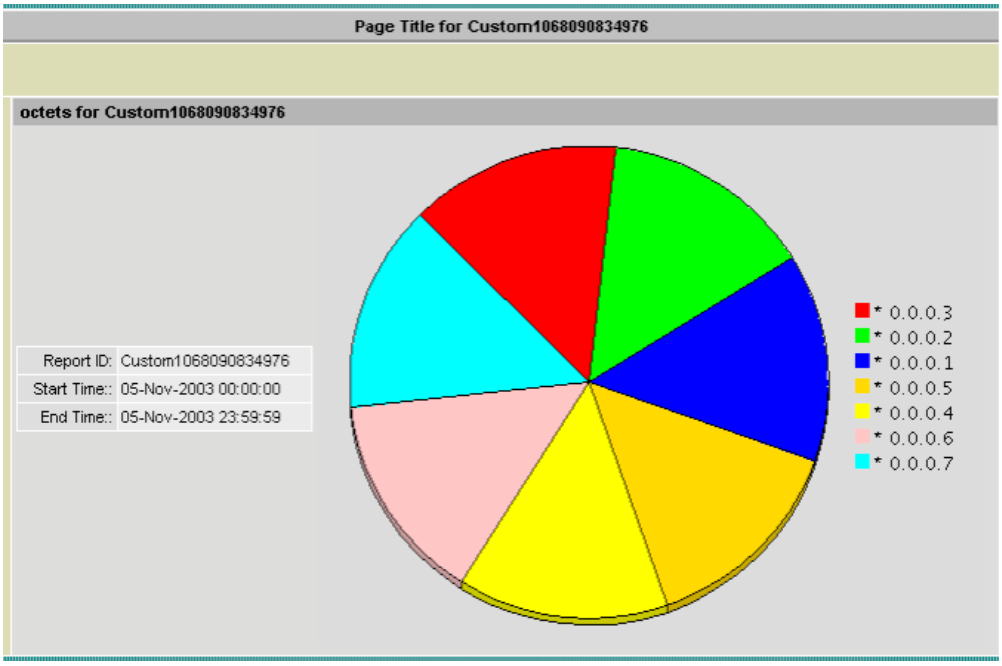


Figure 2-30 Sample Pie Graph

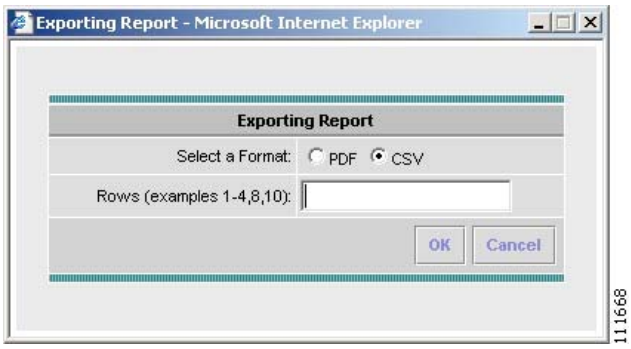


Export and Print

The toolbar icons on the top right of the **Report** window allow you to export and print report data. Click on the export icon to export a report in CSV or PDF format. Click on the print icon to print the report or graph displayed in the current window.

When exporting or printing reports, you can also select which rows to include. For example, the following dialog appears when the export icon is clicked, as shown in Figure 2-31.

Figure 2-31 Exporting Report



Filter

Use the fields at the top right of the report data to filter report data by the key values. The string entered into the text field is treated as a regular expression for matching keys. Click **Filter** to apply the filter. Clear the text field and click **Filter** to return to the original report.

Drill Down

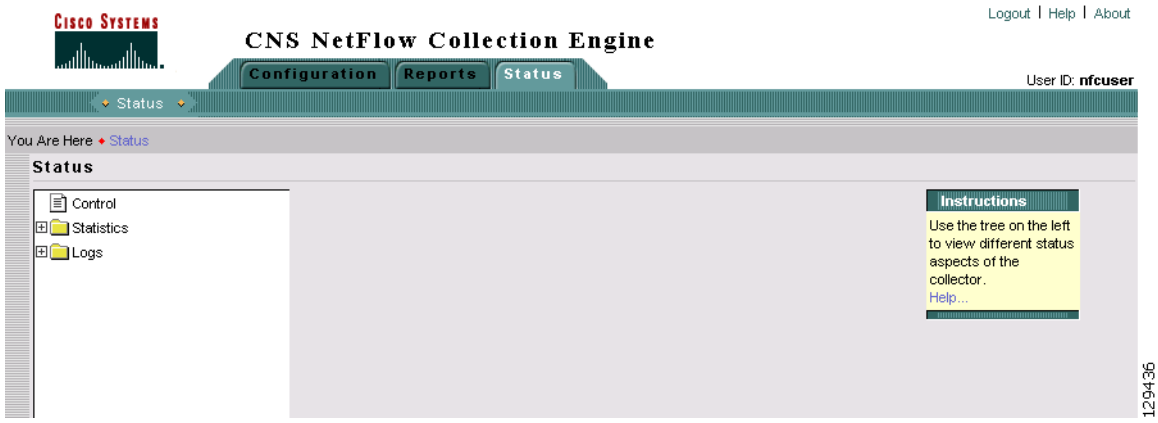
When the original CNS NetFlow Collection Engine output contains more keys than were used to generate a report, you can choose to *drill down* on the data by selecting a row, selecting an addition key, and clicking **Drill Down**. This will generate a new report where the original keys are fixed on the values from the selected row and the drill down key is added to break out the data.

Status

The Status section of the website provides some system health information about the collector. Such information includes running status, flows received statistics, flows missed statistics, and collector logs.

From the **Main** screen you receive upon logging in, click the **Status** tab and you will receive a screen as shown inFigure 2-32.

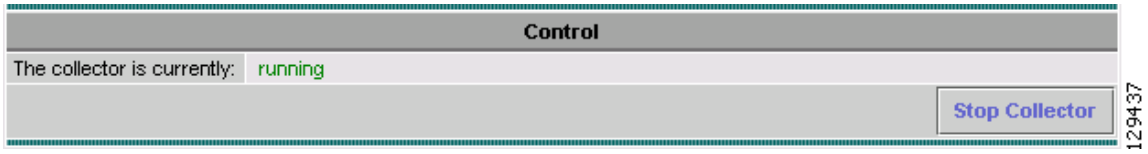
Figure 2-32 Status Screen



Control

Clicking on the **Control** node of the navigation tree displays the running status of the collector, as shown in Figure 2-33. If the collector is running, there will be a button to stop the collector. If the collector is not running, there will be a button to start the collector. The ability to start and stop the collector from the web-based UI is useful for restarting the collector so that configuration changes may take affect. Most operations are not available when the collector is stopped.

Figure 2-33 Control Screen



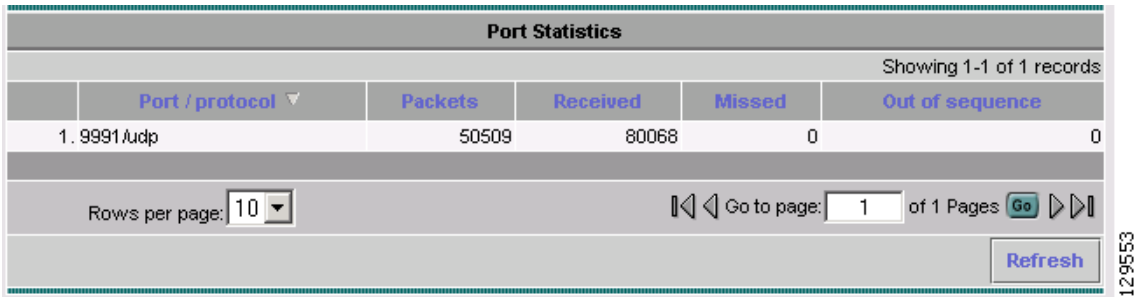
Statistics

The CNS NetFlow Collection Engine collects port and source statistics. The following sections describe Port Statistics and Source Statistics.

Port Statistics

Click on the **Port Statistics** folder of the Statistics navigation tree to display statistics for the ports on which the CNS NetFlow Collection Engine has received data. See Figure 2-34.

Figure 2-34 Port Statistics



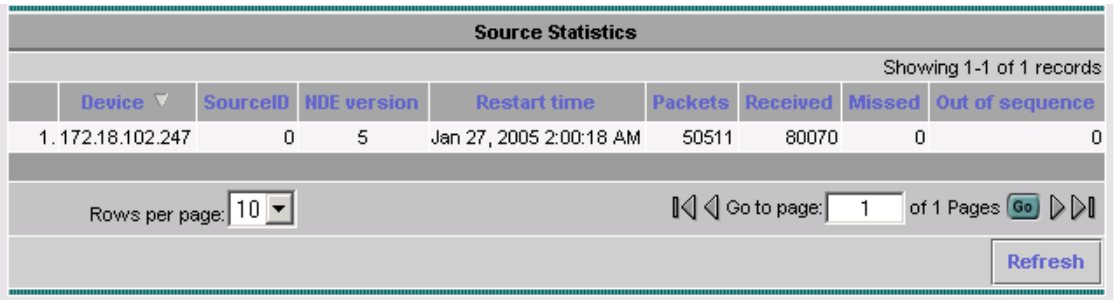
Clicking on **Refresh** updates the statistics shown. The table contains the following fields.

Column	Description
Port/Protocol	Port and protocol for these statistics. For example, 10001/udp.
Packets	Number of packets received.
Received	Number of flows received.
Missed	Number of flows missed (estimate based on sequence number).
Out of sequence	Number of out-of-sequence flows (estimate based on sequence number).

Source Statistics

Click on the **Source Statistics** folder of the Statistics navigation tree to display statistics for the source devices that CNS NetFlow Collection engine has received data from. Source Statistics. SeeFigure 2-35.

Figure 2-35 Source Statistics



Clicking on **Refresh** updates the statistics shown. The table contains the following fields.

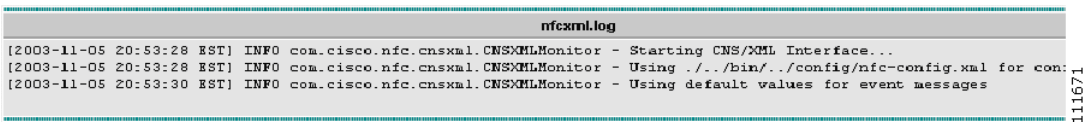
Column	Description
Device	IP address from where the data was received.
SourceID	**source_id** (V9) or **engine_type** and **engine_id** (other versions).
NDE version	Version of data received.
Restart time	Time the device was restarted (estimate based on system uptime in flow data).
Packets	Number of packets received.
Received	Number of flows received.
Missed	Number of flows missed (estimate based on sequence number).
Out of sequence	Number of out-of-sequence flows (estimate based on sequence number).

Each row shown represents a unique combination of the Device, SourceID, and NDE version.

Logs

The logs viewable from the web-based UI are listed under the Logs folder in the navigation tree. Clicking on a specific log loads that log file into the browser window, as shown in Figure 2-36.

Figure 2-36 Viewing Logs in Web-based UI



Troubleshooting

This section contains troubleshooting tips in case you have problems with the web-based UI, as described in Table 2-4.

Table 2-4 Troubleshooting Tips

Problem	Suggestion
The navigation tree is empty	Make sure the collector and CNS/XML interface are running. Make sure the subject parameter of the InitServlet matches that of the CNS/XML interface.
Can not generate a report using a specific aggregator	Only aggregators that have a single ASCII writer can be used to generate reports.

Understanding the CNS NetFlow Collection Engine Data File Format

This chapter tells you how to interpret the data collected and saved in CNS NetFlow Collection Engine data files.

This chapter includes the following sections:

- Data File Directory Structure, page 3-1
- Data Filenames, page 3-4
- Data File Format, page 3-4
- Options Data File Format, page 3-9
- Using the filesready File to Track Data Files, page 3-9.

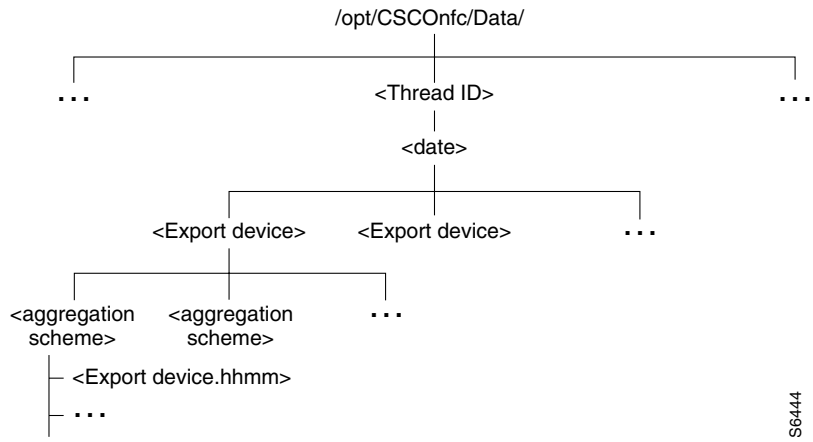
Data File Directory Structure

After you start CNS NetFlow Collection Engine, it begins to collect data based on your aggregation schemes and stores the collected data in data files.

Default Data File Directory Structure

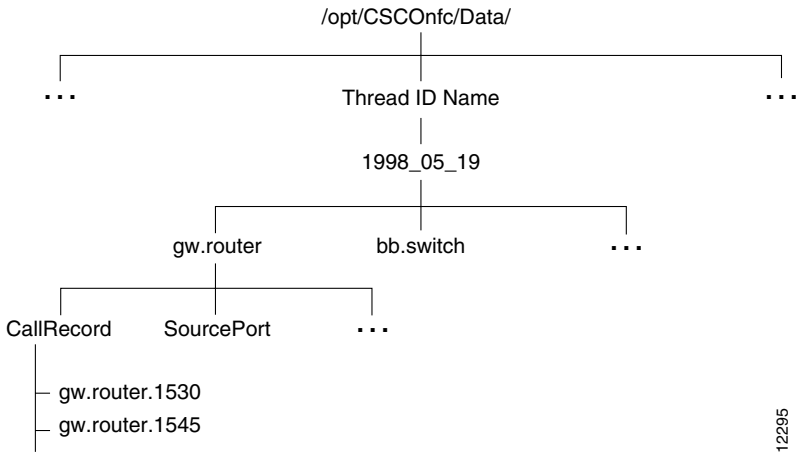
If you specified a custom data file directory path as the **output-base-dir** attribute for a writer associated with an aggregator, the data files are stored in the directory you specified. Otherwise, CNS NetFlow Collection Engine uses the default path, which is **/opt/CSCOnfc/Data,** and the default data file directory structure shown in Figure 3-1.

Figure 3-1 Default Data File Directory Structure



Starting with the specified root directory, a subdirectory is created that identifies the name of the aggregator. A directory is created below that for each day (for example 1999_05_19, as shown in Figure 3-2). Under the date directory, a subdirectory is created for each export device or **nde-source-group** name; and under the export device, there is a subdirectory for each aggregation scheme (for example, **CallRecord** or **SourcePort**). The data files are stored by filename under the aggregation scheme subdirectory. For information on how filenames are formed, see the section "Data Filenames" that follows.

Figure 3-2 Data File Directory Structure Example



Options Data Directory File Structure

NetFlow Data Export Version 9 introduces options data records, a special type of data record based on an options template that provides information about the NetFlow process on the router.

When using NetFlow Data Export (NDE) Version 9 with Cisco CNS NetFlow Collection Engine Release 5.0, options data is made available in much the same way as is aggregated flow data.

The directory structure containing options data reflects the directory structure used to store aggregation data files. The default base directory for options data is **/opt/CSCOnfc/Data/OptionsData**, immediately beneath the default CNS NetFlow Collection Engine data file directory. The NetFlow administrator can specify an alternate path for this in the output-base-dir attribute of the default option-data-writer specified in n**fc-config-predefined.xml**.

Within the **OptionsData** subdirectory, files containing the data are organized in the following format:

```
/opt/CSCOnfc/Data/OptionsData
    <date>
        <export-device>
            <scope-type>
                <scope-value>
                    <export-device>.<timestamp-suffix>
                    <export-device>.<timestamp-suffix>
                    . . .
```

Format conventions for *date*, *export-device*, and *timestamp-suffix* are the same as described in the "Data Filenames" section on page 3-4. In addition, the *filename-includes-date* attribute of *option-data-writer* in **nfc-config-predefined.xml** is used to specify whether a long or short *timestamp-suffix* is appended.

The *scope-type* and *scope-value* are specific to options data. These indicate the part or subset of the NetFlow process or device to which the particular options data applies. In NDE Version 9, scope-type can be one or more of the following values:

- System
- Interface
- Line Card
- NetFlowCache
- Template

If a *scope-type* or *scope-value* is not defined in the *option-data-scope-fields* element in **nfc-config-predefined.xml**, the subdirectory name is written as the integer value of the *scope-type* or *scope-value*. If a *scope-value* has zero length, it is written as the string *NIL*.

Data Filenames

The name given to a data file takes either a long form (<*export-resource-name_yyyy_mm_dd.hhmm*>) or a short form (<*export-resource-name.hhmm*>), depending on the *filename-includes-date* attribute for the writer associated with an aggregator. The short form is the default.

Table 3-1 describes the fields of the data filename format.

Table 3-1 Data Filename Format Fields and Descriptions

Field	Description
name	The domain name system (DNS) name of the export. If the DNS name is not available, the IP address of the export device is used. If an entry for the device is defined in **nde-source-groups** configuration, the group name is written instead.
hhmm	Time when the file was created in hours and minutes.
yyyy_mm_dd	Date in year, month, and day format.

The following are examples of short and long data filenames:

```
gw-router.1530 (short)
gw-router_1996_03_15.1530 (long)
```

Data File Format

Each data file consists of a header followed by one or more aggregation data records. The header contains information about the data records that follow. The fields in each data record correspond with the keys and values in the aggregation scheme associated with this data file. CNS NetFlow Collection Engine Release 5.0.2 supports the following data file formats:

- CSV: Comma or vertical bar-separated header and data records. This format is backwards-compatible with CNS NetFlow Collection Engine Release 3 and 4.
- Mixed: XML header with comma or vertical bar-separated data records.
- XML: XML header and data records. Because of the significant size overhead for XML, it is strongly recommended that you enable compression when XML-only data files are configured.

Note CNS NetFlow Collection Engine Release 5 reporting capability only works with mixed format output files.

Note that starting in CNS NetFlow Collection Engine Release 5.0.3, output records are unsorted by default due to performance considerations. However, sorted output can be enabled through aggregator configuration or through the web-based user interface.

Backwards-Compatible Header

When the global setting *is-output-header-xml* is set to false, output file headers are compatible with Release 4.0. The first line of this header consists of nine field-pairs, each made up of a keyword (in all caps) and its corresponding value (in italic) located to the right of the keyword. The second line of the header begins with the keyword AGGREGATION_DEFINITION and lists the keys and values associated with the output file's aggregation scheme:

```
SOURCE source|FORMAT format|AGGREGATION aggregation|PERIOD period|
STARTTIME time|ENDTIME time|FLOWS flows|MISSED missed|RECORDS records
AGGREGATION_DEFINITION key[|key …]|value[|value …]
```

Note Keywords and their value pairs are separated by either a vertical bar (|) or a comma (,). You can choose between a vertical bar or comma by specifying the global setting output-field-delimiter.

See Table 3-2 for descriptions of the fields in the data file header section.

Table 3-2 Data File Header Keywords and Descriptions

Keyword	Description
SOURCE	Identifies the source of the NetFlow export traffic summarized in this data file. The label can be the IP address, in dotted decimal format, or an ASCII name. If you are using the **nde-source-groups** feature, the label is the group name specified in the **nde-source-groups** configuration element.
FORMAT	Tracks the version of the data file. Because CNS NetFlow Collection Engine, Release 4.0 uses tag **2**, that is the value used here.
AGGREGATION	The name of the aggregation scheme used to create this data file.
PERIOD	The data collection period, specified in minutes. Under some circumstances, CNS NetFlow Collection Engine might generate a data file before the current data collection period expires. In such a case, CNS NetFlow Collection Engine adds the keyword **PARTIAL** to the filename, and the **PERIOD** field in the header is identified as **PERIOD PARTIAL**. For more information on partial data files, see the "Partial Data Files" section on page 3-8.
STARTTIME	The time in Coordinated Universal Time (UTC) seconds when this data collection period began.
ENDTIME	The time in UTC seconds when this data collection period ended.
FLOWS	The total number of NetFlow export records that are aggregated in this data file.
MISSED	The number of flow records that CNS NetFlow Collection Engine should have received but did not. The MISSED value is derived from the sequence numbers (where present) in each packet. If the only data aggregated into a data file is from a V1 NetFlow export datagram or a V7 NetFlow export datagram with shortcut mode turned on, the MISSED field in the header contains -1 as the value.
RECORDS	The count of the aggregation records present in this data file.
SAMPLEMODE (optional)	The numeric ID of the sampling scheme enabled on the exporting device. This keyword only appears when you specify the aggregator to output sampling information. See Chapter 4, "Customizing the CNS NetFlow Collection Engine." for more information.
SAMPLEINTERVAL (optional)	The sampling interval of the sampling scheme enabled on the exporting device. This keyword only appears when you specify the aggregator to output sampling information.

The second line of the header lists the key and value fields defined by the aggregation scheme associated with the file. For example, the **CallRecord** aggregation scheme defines six key fields (srcaddr, dstaddr, srcport, dstport, prot, tos) and six value fields (pkts, octets, flows, starttime, endtime, activetime). The aggregation definition line for **CallRecord** would be written as follows:

```
AGGREGATION_DEFINITION srcaddr|dstaddr|srcport|dstport|prot|tos|
pkts|octets|flows|starttime|endtime|activetime
```

Note In the aggregation definition, as in the header, keywords and their value pairs are separated by either a vertical bar (|) or a comma (,) as determined by the global setting *output-field-delimiter*.

Data File Example

The following data file example for the **CallRecord** aggregation scheme shows the data file header and the first two aggregation definition data records.

```
SOURCE 192.1.134.7|FORMAT 2|AGGREGATION CallRecord|PERIOD 15|STARTTIME
881972378|
ENDTIME 881973278|FLOWS 59709|MISSED 0|RECORDS 2345
AGGREGATION_DEFINITION
srcaddr|dstaddr|srcport|dstport|prot|tos|pkts|octets|flows|starttime|endtim
e|activetime
171.69.1.17|172.23.34.36|2963|6000|6|114|2|176|1|768550628|768550628|0
171.69.1.23|171.69.25.133|2972|6500|17|0|3|172|1|768520516|768520520|4135
.
.
.
```

In the **CallRecord** aggregation scheme, the key portion of the data record is the first six fields and consists of the following aggregation fields:

```
srcaddr|dstaddr|srcport|dstport|prot|tos
```

For example, the first six fields in the second data record from the example above are:

```
171.69.1.23|171.69.25.133|2972|6500|17|0
```

These fields are described Table 3-3.

Table 3-3 Data File Fields

Field	Description
171.69.1.23	Source IP address (srcaddr).
171.69.25.133	Destination IP address (dstaddr).
2972	Source port (srcport).
6500	Destination port (dstport).
17	Protocol byte (prot).
0	Type of service (ToS).

The value portion is the last six fields and consists of the following aggregation values:

```
pkts|octets|flows|starttime|endtime|activetime
```

For example, the last six fields in the second data record from the example above are:

```
3|172|1|768520516|768520520|4135
```

XML Header Format

When the global setting is-output-header-xml is set to *true*, output files are written with a new XML header that contains additional information. Using the new XML format, the example above would appear as follows:

```
<?xml version="1.0"?>
<nfc-output-header xmlns="http://www.cisco.com/nfc/output"
version="1.0">
    <source>192.1.134.7</source>
    <aggregator>CallRecordTest</aggregator>
    <aggregation-scheme name="CallRecord">
        <key name="srcaddr" type="ipaddress"/>
        <key name="dstaddr" type="ipaddress"/>
        <key name="srcport" type="integer"/>
        <key name="dstport" type="integer"/>
        <key name="prot" type="integer"/>
        <key name="tos" type="integer"/>
        <value name="pkts" type="integer"/>
        <value name="octets" type="integer"/>
        <value name="flows" type="integer"/>
        <value name="starttime" type="utc"/>
        <value name="endtime" type="utc"/>
        <value name="activetime" type="integer"/>
    </aggregation-scheme>
    <delimiter>|</delimiter>
    <period-minutes>15</period-minutes>
    <starttime>881972378</starttime>
    <endtime>881973278</endtime>
    <flows>59709</flows>
    <missed>0</missed>
    <records>2345</records>
</nfc-output-header>
171.69.1.17|172.23.34.36|2963|6000|6|114|2|176|1|768550628|768550628|0
171.69.1.23|171.69.25.133|2972|6500|17|0|3|172|1|768520516|768520520|4135
.
.
```

In addition to the information contained in the old header format, the new XML header adds the type of each key and value field, and the name of the aggregator that generated the file. Key and value types supported are *integer*, *ipaddress*, *string*, and *utc* (seconds since January 1, 1970).

The XML header contains the following elements.

Element	Description
source	IP address of the source router, or nde-source-group name.
aggregator	Name of the aggregator that produced the file.
aggregation-scheme	Name of the aggregation scheme associated with this aggregator.
key	Name and type of the keys in the aggregation scheme. Supported types: integer, ipaddress, string, utc.
value	Name and type of the values in the aggregation scheme. Supported types: integer, ipaddress, string, utc.
delimiter	Output field delimiter.

Element	Description
period-minutes	Aggregation time period in minutes.
starttime	UTC time when the data collection period began.
endtime	UTC time when the data collection period ended.
flows	Number of flows aggregated.
missed	Number of flows missed during aggregation.
records	Number of output records (lines in the file not including the header).

An XML parser is not required for parsing the XML header. Because the header can contain a variable number of lines, standard UNIX utilities such as awk can be used to separate the header and data records. The XML header will always end with a line containing only the tag **</nfc-output-header>**, making it straightforward to identify the end of the header and the beginning of data records.

Partial Data Files

Under normal circumstances, CNS NetFlow Collection Engine generates a data file periodically as determined by the period specified in aggregator definition. See the"Creating an Aggregator" section on page 4-16.

Under certain circumstances, CNS NetFlow Collection Engine might be forced to generate a data file before the current data collection period expires. Such a data file is called a partial data file because it does not represent data collected for the entire defined collection period. This can occur when an aggregator definition is modified through the web UI or the CNS XML interface.

The data in a partial data file is valid data; the file just does not represent a full data collection period and is differentiated to prevent data statistics from being distorted by comparing data from full and partial periods.

Because the current data collection period has not expired when the file is written, CNS NetFlow Collection Engine generates and marks the data files differently: the keyword **partial-** and the UTC time stamp are added to the data filename as a suffix, and the **PERIOD** field in the header is identified as **PARTIAL**.

In the following two data file examples, the first example shows a complete data file, and the second example shows a partial data file (using a different aggregation scheme).

```
SOURCE gw.router|FORMAT 2|AGGREGATION Protocol|PERIOD 10|STARTTIME
923416099|
ENDTIME 923416699|FLOWS 2868330|MISSED 154590|RECORDS 1
AGGREGATION_DEFINITION protocol|pkts|octets|flows
ICMP|2868330|2868330|2868330

SOURCE gw.router|FORMAT 2|AGGREGATION DestPort|PERIOD PARTIAL|
STARTTIME 923419273|ENDTIME 923419607|FLOWS 4467930|MISSED 0|RECORDS 250
AGGREGATION_DEFINITION dstport|pkts|octets|flows
1|17872|2626340864|17872
2|17872|2626340864|17872
..
..
```

Note The first data file generated by CNS NetFlow Collection Engine can contain less data than what would be collected during the full aggregation period and not be labeled as PARTIAL if the global setting start-output-at-top-of-the-hour is true. Refer to the "Global Settings" section on page 4-25 for information on global settings.

Options Data File Format

The following is an example options data file:

```
SOURCE 172.23.3.167|SCOPE_TYPE interface|SCOPE_VALUE 2
SAMPLING_INTERVAL 100
SAMPLING_ALGO 1
```

The first line in this example is a header consisting of the source device, scope type, and scope value from the option information. Subsequent lines contain the options in the options data messages received from the device during this aggregation period.

Option names and scope type names are obtained from the configuration file **nfc-config-predefined.xml**. If a name mapping does not appear in the file, the decimal value for the option type or scope type is written.

IP address values such as the source ip address are either written in dotted-decimal or integer format as determined by the *ipaddress-output-format* setting (as with aggregated data); other values that are 1 to 8 bytes in length are in integer format; and values greater than 8 bytes in length are output as hexadecimal digits proceeded by **0x**.

As with aggregated data, the separator on the first line is either a vertical bar or comma as determined by the *output-format-delimiter* global setting.

Using the filesready File to Track Data Files

CNS NetFlow Collection Engine periodically appends the absolute path names of data files that it has generated to a list in a log file named **filesready**. CNS NetFlow Collection Engine identifies the file with the time stamp *YYYY_MM_DD*, where *YYYY_MM_DD* represents the year, month, and day. The **filesready** file is located with the other log files in the **$NFC_DIR/logs** directory.

Typically, a client application reads this file every *n* minutes, processes it to determine the names of any newly added data files, and then retrieves those new data files. Alternately, starting in Release 5.0, a client can listen for filesready file update events on the CNS Integration Bus. See the "Event Service" section on page 5-3 for additional details.

After it finishes writing a new data file, CNS NetFlow Collection Engine appends the absolute path name of the new data file onto the list in the **filesready** file. The **filesready** file contains a header that indicates the format version in use. The following example shows the header, contents, and organization of a typical **filesready** file.

```
FORMAT A
/opt/CSCOnfc/Data/1998_02_11/171.71.34.79/Protocol/171.71.34.79.2135
/opt/CSCOnfc/Data/1998_02_11/171.71.34.79/DetailASMatrix/171.71.34.79.2136
/opt/CSCOnfc/Data/1998_02_11/171.71.34.79/Protocol/171.71.34.79.2136
/opt/CSCOnfc/Data/1998_02_11/171.71.34.79/DetailASMatrix/171.71.34.79.2137
/opt/CSCOnfc/Data/1998_02_11/171.71.34.79/Protocol/171.71.34.79.2137
/opt/CSCOnfc/Data/1998_02_11/171.71.34.79/DetailASMatrix/171.71.34.79.2138
/opt/CSCOnfc/Data/1998_02_11/171.71.34.79/Protocol/171.71.34.79.2138
```

Options Data Filesready File

As with aggregated data files, the CNS NetFlow Collection Engine periodically appends the names of new options data files to a file in the CNS NetFlow Collection Engine logs directory. The name of this file is **$NFCDIR/logs/optionsdata-filesready.<YYYY_MM_DD>**.

Each line in the file is the complete pathname of an options data file. As with the aggregated data **filesready** file, the first line may in the future contain a FORMAT specification. However, this will not appear in CNS NetFlow Collection Engine, Release 5.0.

Customizing the CNS NetFlow Collection Engine

This chapter describes how to customize CNS NetFlow Collection Engine operations using thread, filter, and protocol definitions, lists of port and autonomous system numbers, and other CNS NetFlow Collection Engine configuration parameters.

This chapter includes the following sections:

- CNS NetFlow Collection Engine Configuration and Resource Files, page 4-1
- CNS NetFlow Collection Engine Data Collection and Aggregation, page 4-2
- Tuning Memory Usage, page 4-26
- Managing Disk Space, page 4-28
- NetFlow Collection Engine Logger Configuration, page 4-29

CNS NetFlow Collection Engine Configuration and Resource Files

The process of customizing CNS NetFlow Collection Engine operation involves changes and additions to one or more of the following CNS NetFlow Collection Engine configuration and resource files located in the **NFC_DIR/config** directory.

You can use a text editor to change any of these files. You can also use the CNS NetFlow Collection Engine UI to update the CNS NetFlow Collection Engine configuration. For details on the use of the CNS NetFlow Collection Engine UI, see Chapter 2, “Using the CNS NetFlow Collection Engine User Interface,” for more information.

XML Configuration and Schema

A basic working knowledge of XML and XML Schema is needed for editing CNS Netflow Collection Engine, Release 5.0 configuration. An overview of XML is beyond the scope of this document. However, there are a number of resources for this available freely on the internet as well as a large number of books on the topic. A good starting point might be the following W3C pages:

http://www.w3.org/XML

http://www.w3.org/XML/Schema.

The following XML and related schema files in the directory **NFC_DIR/config** work together to define configuration information for CNS Netflow Collection Engine:

- **nfc-config.xml**
- **nfc-config-predefined.xml**
- **nfc-config.xsd**
- **nft-types.xsd**

Configuration information for the core collector application is kept in two files:

- **nfc- config.xml**
- **nfc-config-predefined.xml**

The file **nfc-config-predefined.xml** contains default values and values that the customer typically does not change. You can apply additions and changes to **nfc-config.xml**, which CNS Netflow Collection Engine merges with the predefined file at runtime. If a definition appears in both files, with your changes in **nfc- config.xml** taking precedence.

The XML schema for the types referenced in **nfc-config.xml** and **nfc-config- predefined.xml** is contained in the file **nfc-types.xsd**. The overall structure (order) of document elements in these XML files is defined in **nfc-config.xsd**. An advanced user can refer to these files to troubleshoot document XML format errors when making configuration changes. Logs resulting from this sort of error may reference lines in any of these files.

The configuration files for CNS Netflow Collection Engine 4.0 and earlier are no longer supported in CNS Netflow Collection Engine 5.0. A tool is provided that helps customers migrate their configuration. For information on this tool, see to Appendix G, “CNS NetFlow Collection Engine Migration Tools.”

CNS NetFlow Collection Engine Data Collection and Aggregation

CNS NetFlow Collection Engine collects and summarizes (aggregates) data into data files based on user-defined criteria specified in a CNS NetFlow Collection Engine aggregator. An aggregator is a task defined by a set of user-configurable attributes that specify how CNS NetFlow Collection Engine summarizes the traffic flows stored on the workstation.

Overview of CNS NetFlow Collection Engine 5.0 Configuration Information

Table 4-1 briefly describes the top-level document elements that can appear in the configuration files **nfc-config.xml** and **nfc-config-predefined.xml**. You can not change elements that are indicated as not editable, unless specifically noted.

Table 4-1 CNS NetFlow Collection Engine Configuration Elements

Element	Editable	Purpose
global	yes	Settings that affect the operation of the collector as a whole. Many of these settings appeared in the **nf.resources** file in releases prior to 5.0.
field-info	yes	The name, type ID, content type, and alternate name for each field. Corresponds with entries in **nfknown.typefile** in release 4.0. You can update field information to add new field aliases and to add field definitions as new router features introduce new V9 fields.
fixed-flow-packet-types	no	Describes the structure of all pre-NDE V9 packet types (field offsets, header size, etc.). You can not typically change the contents of this element, although as new NDE V8 router-based aggregation schemes are added, additional packet descriptions can be added.
flow-interpreters	no	Defines an interpreter for each fixed-flow-packet-type as well as for NDE V9.
key-builders	yes	Key definitions referenced by aggregation schemes.
value-builders	yes	Value definitions referenced by aggregation schemes.
option-data-map	yes	Defines option data map entries that are cached for use by option data key builders.
filters	yes	Filter definitions referenced by aggregators.
aggregation-schemes	yes	Aggregation scheme definitions, such as which keys and values are aggregated and output; referenced by aggregators.
binary-output-mapping-info	no	Describes how aggregation output is mapped to CNS NetFlow Collection Engine 4.0.1 compatible binary output format.
aggregators	yes	Aggregator definitions (*threads* in CNS NetFlow Collection Engine versions prior to Release 5.0).
nde-source- groups	yes	Maps one or more router IP addresses and/or hostnames to a group name. In CNS NetFlow Collection Engine versions prior to Release 5.0 this was the ROUTER_GROUPAME resource.
nde-source- access-list	yes	Access list for NDE sources. In CNS NetFlow Collection Engine versions prior to Release 5.0 this was the ACCEPT_PACKETS_FROM resource.
flow-readers	no	Flow readers, one per NDE protocol. In CNS NetFlow Collection Engine, Release 5.0, the only protocol supported is udp. Other protocols such as sctp or tcp can be added in subsequent releases.

Table 4-1 CNS NetFlow Collection Engine Configuration Elements (continued)

Element	Editable	Purpose
option-data-listeners	no	Option data processors. In CNS NetFlow Collection Engine, Release 5.0, the only option data processor produces output that is backwards compatible with CNS NetFlow Collection Engine, Release 4.0 option data output.
cns-xml-interface	no	Contains settings for the remote configuration interface for external clients.
event-service	no	Defines supported transports for CNS NetFlow Collection Engine events and related options. Syslog and CNS Integration Bus are the formats supported in CNS NetFlow Collection Engine, Release 5.0. Although this is typically not changed, depending on how the CNS Integration Bus is used in the network additional options might be needed.
disk-usage-monitor	yes	Lists file systems to be monitored for available space.

The order of these elements in the XML is determined by the schema file **nfc-config.xsd**.

Fields

Fields represent individual items of data sent by a router in a NetFlow flow. Although the user will not typically add or change field definitions, they are important to consider since they are the building blocks upon which the keys and values referenced by aggregation schemes are built.

A field contains no information about what CNS NetFlow Collection Engine does with the data in a flow; it is simply an identifier for data of interest, and indexes *template* information for the flow that indicates the field's offset and length within an NDE flow. In CNS NetFlow Collection Engine, Release 5.0, even for pre- NDE V9 packets, templates are defined internally that allow pre-V9 and V9 packets to be treated the same way.

Fields are declared in **nfc-config-predefined.xml** as follows:

```
<field-info>
    <fields>
        <field id="1" name="IN_BYTES">
            <alias name="octets"/>
        </field>
        <field id="2" name="IN_PKTS">
            <alias name="pkts"/>
        </field>
        <field id="4" name="PROTOCOL">
            <alias name="prot"/>
        </field>
        <!-- etc. -->
    </fields>
</field-info>
```

The fields element contains a list of all fields that are recognized by CNS NetFlow Collection Engine. Field aliases define alternate names for the subset of fields that appear in V1-V8 NDE that were used in pre-defined aggregation schemes in versions of CNS NetFlow Collection Engine prior to 5.0. An aggregation scheme can reference either the field name or its alias.

Keys and Values

An aggregation scheme consists of *keys* and *values*. Within an aggregation period, each value within flows having the same set of keys is aggregated (typically summed) together with the corresponding values from earlier matching flows within an aggregation period.

Fields are not referenced directly by an aggregation scheme; instead, a *key builder* or *value builder* references a field, and one or more aggregation schemes references the builder.

Whereas a field is simply an identifier for data within a NetFlow flow, key builders and value builders assign additional semantic meaning to a field: its type (integer, string, ipaddress, UTC time), output format, column name in output, and possibly operations or transformations performed on the data.

Note that in special cases it is possible for a builder to not be associated with any field (such as flow count), or for a builder to combine and transform data from more than one field (such as multi-field map).

The following sections give a brief description for each type of key and value builder that can be created and its configurable attributes and elements. There are predefined examples of most of these types in **/opt/CSCOnfc/config/nfc-config-predefined.xml**; the exact syntax for defining each type appears in the XML schema in **/opt/CSCOnfc/config/nfc-types.xsd**. The web-based user interface makes it somewhat easier to create new builder instances. Refer to the "Key Builders" section on page 2-8 and the "Value Builders" section on page 2-14 for additional information.

Key Builder Types

address-range-map-key

The **address-range-map-key** builder reads an IP address and performs a map lookup to obtain a string value. If no mapping is found, a default label is used if one is specified; otherwise a string representation of the value itself is output.

Key Builder	Description
name	Column name in output; defaults to the field ID if not specified.
field	ID of the field to lookup from flows.
default-label	Output value if no mapping result is found; otherwise if not specified the value itself is output
is-null-allowed	If set to *false* (default) and a flow does not contain field, an error is logged. I f set to *true*, the output value is empty and no error is logged.
ranges	Contains one or more range and/or value definitions.

Any number and combination of address values and ranges of values can be specified. More than one value or range can be associated with one label, however values and ranges cannot overlap. A range is defined with the following elements and attributes:

Element	Attributes
minimum	Minimum value in the range.
maximum	Maximum value in the range.
label	Output value associated with the range.

A value element itself contains an integer value, and has the following attribute:

Element	Attributes
label	Output value associated with the range.

bgp-attr-post-agg-key, bgp-attr-key

The **bgp-attr-post-agg-key** and **bgp-attr-key** builders perform a longest match lookup on an address obtained from a flow against address prefixes advertised by the originator of the flow, which is also a peer of the passive CNS NetFlow Collection Engine BGP Peer. The lookup result is BGP-4 attribute as described in RFC-1771. If the lookup fails, the result is an empty string. Refer to the"BGP Peer" section on page 5-8 for information about configuring and starting the CNS NetFlow Collection Engine BGP Peer.

The **bgp-attr-post-agg-key** builder performs a post-aggregation lookup once at the end of each aggregation period. The **bgp-attr-key** builder performs a lookup as each flow arrives, which reduces lookup latency but results in a very significant performance penalty. Because of this performance penalty, the **bgp-attr-post-agg-key should** be used under most circumstances.

Note When using the bgp**-attr-post-agg-key** builder, the containing aggregation scheme must contain the builder referenced by *address-key* in the builder configuration.

Key Builder	Description
name	Column name in output.
address-field	ID of the key builder that obtains an for which the attribute lookup is performed.
attribute	Integer attribute ID or one of the attribute names as defined in RFC-1771: ORIGIN AS_PATH NEXT_HOP MED LOCAL_PREF ATOMIC_AGGREGATE AGGREGATOR COMMUNITY ORIGINATOR_ID CLUSTER_LIST

bgp-complete-as-path-post-agg-key, bgp-complete-as-path-key

The **bgp-complete-as-path-post-agg-key** and **bgp-complete-as-path-key** builders perform a longest match lookup on source and destination addresses obtained from a flow against address prefixes advertised by the originator of the flow, which is also a peer of the passive CNS NetFlow Collection Engine BGP Peer. The results is a complete AS path from source to destination that is constructed from two separate lookups. The accuracy of the result assumes a symmetrical route to and from the source and the peer. If the lookup fails, the result is an empty string.

The **bgp-complete-as-path-post-agg-key** builder performs a post-aggregation lookup once at the end of each aggregation period. The **bgp-complete-as-path-key** builder performs a lookup for as each flow arrives, which reduces lookup latency but results in a very significant performance penalty. Because of this performance penalty, the **bgp-complete-as-path-post-agg-key** variant should be used under most circumstances.

Note When using the **bgp-complete-as-path-post-agg-key** builder, the containing aggregation scheme must also contain the builders referenced by *srcaddr-key* and *dstaddr-key* in the builder configuration.

Key Builder	Description
name	Column name in output.
srcaddr-key	ID of key builder that obtains the source address.
dstaddr-key	ID of key builder that obtains the destination address.

bit-field-key

The **bit-field-key** builder reads an integer value and extracts a specified number of bits starting at a specified offset within the integer. Outputs the bit range as an integer value.

Key Builder	Description
name	Column name in output.
field	ID of the address field to lookup from flows.
least-significant-bit	Offset from zero of the least significant bit to include.
num-bits	Number of bits to include.
format	Either *decimal* (default) or *hex*.
is-null-allowed	If set to *false* (default) and a flow does not contain either field, an error is logged. I f set to *true*, the output value is empty and no error is logged.

inetaddress-key

The **inetaddress-key** builder reads and outputs an IP address. Both IPV4 and IPV6 addresses are supported.

Key Builder	Description
name	Column name in output; defaults to the field ID if not specified.
field	ID of the field to lookup from flows.
format	One of *standard-notation* (default), *hostname*, or *integer*. The *integer* format option will be removed in a future release and is not valid for IPV6 addresses.
is-null-allowed	If set to *false* (default) and a flow does not contain field, an error is logged. I f set to *true*, the output value is empty and no error is logged.

If *format* is specified as *hostname*, a DNS lookup is performed at output time to determine the hostname associated with the address. If the lookup fails, the IP address is returned instead as a string. Hostname caching is performed by the Java Runtime Environment, which has tunable parameters for the cache. See *InetAddress Caching* at http://java.sun.com/j2se/1.4.2/docs/api/java/net/InetAddress.html for additional information on how hostnames are cached by the Java Runtime Environment, and on how caching behavior can be tuned.

integer-key

The **integer-key** builder reads and outputs an integer.

Key Builder	Description
name	Column name in output; defaults to the field ID if not specified.
field	ID of the address field to lookup from flows.
format	Either *decimal* (default) or *hex*.
is-null-allowed	If set to *false* (default) and a flow does not contain either field, an error is logged. I f set to *true*, the output value is empty and no error is logged.

integer-range-map-key

The **integer-range-map-key** builder reads an integer and performs a map lookup to obtain a string value. If no mapping is found, a default label is used if one is specified; otherwise a string representation of the value itself is output.

Key Builder	Description
name	Column name in output; defaults to the field ID if not specified.
field	ID of the field to lookup from flows.
default-label	Output value if no mapping result is found; otherwise if not specified the value itself is output
is-null-allowed	If set to *false* (default) and a flow does not contain either field, an error is logged. I f set to *true*, the output value is empty and no error is logged.
ranges	Contains one or more range and/or value definitions.

Any number and combination of integer values and ranges of values can be specified. More than one value or range can be associated with one label, however values and ranges cannot overlap. A range is defined with the following elements and attributes:

Element	Attributes
minimum	Minimum value in the range.
maximum	Maximum value in the range.
label	Output value associated with the range.

A value element itself contains an integer value, and has the following attribute:

Element	Attributes
label	Output value associated with the range.

interface-name-key

The **interface-name-key** builder maps an interface index in a flow to the interface name obtained via an SNMP query to the device. The query result is cached in order to improve lookup performance.

Key Builder	Description
name	Column name in output; defaults to the field ID if not specified.
field	ID of the interface field to lookup from flows.
is-null-allowed	If set to *false* (default) and a flow does not contain either field, an error is logged. I f set to *true*, the output value is empty and no error is logged.

mask-field-inetaddress-key

The **mask-field-inetaddress-key** builder reads both an IP address and mask value from a flow, and applies the mask to the address. The masked value is output.

Key Builder	Description
name	Column name in output.
address-field	ID of the address field to lookup from flows.
mask-field	ID of the mask field to lookup from flows.
format	One of *standard-notation* (default), *hostname*, or *integer*. The *integer* format option will be removed in a future release and is not valid for IPV6 addresses.
is-null-allowed	If set to *false* (default) and a flow does not contain either field, an error is logged. I f set to *true*, the output value is empty and no error is logged.

masked-inetaddress-key

The **masked-inetaddress-key** builder reads an IP address and applies a specified mask value. Note the difference between this builder and **mask-field-inetaddress-key**, which obtains the mask from the flow

Key Builder	Description
name	Column name in output.
field	ID of the address field to lookup from flows.
mask	Integer mask value to apply.
format	One of *standard-notation* (default), *hostname*, or *integer*. The *integer* format option will be removed in a future release and is not valid for IPV6 addresses.
is-null-allowed	If set to *false* (default) and a flow does not contain field, an error is logged. I f set to *true*, the output value is empty and no error is logged.

multi-field-map-key

The **multi-field-map-key** builder combines the output from a combination other key builders to produce a mapping result. For example, a generic implementation of CNS NetFlow Collection Engine 3.x/4.0-style protocol mapping whose value depends on the protocol byte, source port, and destination port is supplied in predefined configuration.

See "Creating a Multi-Field Map" section on page 4-21 for additional information about multi-field maps.

option-data-key

The **option-data-key** builder specifies a map lookup from the option data map, described in the "Creating an Option Data Map" section on page 4-22, using the combination of one or more fields in a flow. One or more key fields are configured in an instance of the option-data-key; these fields are extracted from a traffic flow record and used to perform the option data map lookup. The result of the lookup is the output value for this key builder. If no match is found in the option data map, the value displayed in aggregation output is an empty string.

Key Builder	Description
name	Column name in output.
option-data-map-entry	ID of the option data map entry containing data of interest.
keys	One or more IDs of key builders that produce the key set for performing the option data map lookup.

string-key

The **string-key** builder reads and outputs a UTF-8 string value.

Note String values in flow data are supported only in NetFlow Data Export version 9.

Key Builder	Description
name	Column name in output; defaults to the field ID if not specified.
field	ID of the field to lookup from flows.
is-null-allowed	If set to *false* (default) and a flow does not contain either field, an error is logged. I f set to *true*, the output value is empty and no error is logged.

Value Builder Types

active-time-value

The **active-time-value** builder calculates and outputs the time difference in seconds between the start-time-field value reported in the first flow in the aggregation period and the end-time-value reported in the last flow in the aggregation period.

Key Builder	Description
name	Column name in output; defaults to the field ID if not specified.
start-time-field	ID of the start time field to lookup from flows.
end-time-field	ID of the end time field to lookup from flows.

end-time-value

The **end-time-value** builder reads and outputs the time in UTC seconds reported in the last flow processed in the aggregation period.

Key Builder	Description
name	Column name in output; defaults to the field ID if not specified.
end-time-field	ID of the end time field to lookup from flows.

flow-count-value

The **flow-count-value** builder increments a count by one for each flow received and outputs the total.

Key Builder	Description
name	Column name in output; defaults to the field ID if not specified.

max-burst-rate-value

The **max-burst-rate-value** builder maintains the largest value of a rate calculated for any each flow in an aggregation period. Reads the count in a flow (typically octets) and divides that by the flow active time in seconds.

Key Builder	Description
name	Column name in output; defaults to the field ID if not specified.
start-time-field	ID of the start time field to lookup from flows.
end-time-field	ID of the end time field to lookup from flows.
octets-field	ID of the count field to lookup from flows.

rate-value

The **rate-value** builder references the **sum-value** or **flow-count-value** builder to accumulate a count, and divides the final count by the number of seconds or minutes in the aggregation period.

Key Builder	Description
name	Column name in output; defaults to the **quantity-value-builder** ID appended with *per-second* or *per-minute* if not specified.
quantity-value-builder	ID of a value builder that keeps a count of the instances of **sum-value** or **flow-count-value** builders.
unit	Specifies if the final count is divided by *seconds* or *minutes*.

start-time-value

The **start-time-value** builder reads and outputs the time in UTC seconds reported in the earliest flow processed in the aggregation period.

Key Builder	Description
name	Column name in output; defaults to the field ID if not specified.
start-time-field	ID of the start time field to lookup from flows.

sum-value

The **sum-value** builder reads an integer from a flow, adds it to a running total, and outputs the total.

Key Builder	Description
name	Column name in output; defaults to the field ID if not specified.
field	ID of the field to lookup from flows.

Key and Value Builder Data Types

The following tables display the data type for each key and value builder. This data type is include in the key or value description in the XML header of the CNS NetFlow Collection Engine output files.

Table 4-2 describes key builders you can use for defining keys in an aggregation scheme.

Table 4-2 Aggregation Scheme Key Builders

Key Builder	Data Type
integer-key	integer
inetaddress-key	ipaddress
mask-field- inetaddress-key	ipaddress
masked- inetaddress-key	ipaddress

Table 4-2 Aggregation Scheme Key Builders (continued)

Key Builder	Data Type
address-range- map-key	string
bgp-attr-post-agg-key, bgp-attr-key	string
bgp-complete-as-path-post-agg-key, bgp-complete-as-path-key	string
bit-field-key	integer
interface- name-key	string
integer-range- map-key	string
mpls-exp-key	integer
multi-field- map-key	string
option-data-key	Depends on the value type of the referenced option-data-map-entry
string-key	string

Table 4-3 displays value builders you can use for defining values in an aggregation scheme.

Table 4-3 Aggregation Scheme Value Builders

Value Builder	Data Type
active-time-value	integer
end-time-value	UTC
flow-count-value	integer
max-burst-rate-value	integer
rate-value	integer
start-time-value	UTC
sum-value	integer

For each data row in CNS NetFlow Collection Engine output files, the corresponding aggregation scheme references instances of key and value builders declared in <*key-builders*> and <*value-builders*>, one for each column of data. A key or value builder always contains an ID attribute used to reference the builder instance from aggregation schemes. In most cases a key or value builder references a <*field*> element that identifies which field in a flow contains the data of interest. The column name in output can be specified in a <*name*> element; otherwise, unless the field name is ambiguous, the output column name defaults to the field name.

Note A key or value builder can be referenced by several different aggregation schemes. For example, the srcaddr key appears in many different aggregation schemes. Moreover, keys and values and therefore aggregation schemes are independent of any particular version of NDE, as long as a particular flow contains the field that the builder references.

To illustrate this, the following is an example of instances of a key and value builder defined in **nfc-config-predefined.xml**:

```
<key-builders>
    <inetaddress-key id="srcaddr-key">
        <field>srcaddr</field>
    </inetaddress-key>
    [additional key builders not shown]
</key-builders>

<value-builders>
    <sum-value id="packet-count-value">
        <field>pkts</field
    </sum-value>
    [additional value builders not shown]
</value-builders>
```

In this example, the key *srcaddr-key* is declared as an instance of the key builder *inetaddress-key*, and references the field *srcaddr*. Because a *name* element is not specified, the column name in output defaults to *srcaddr*; all instances of *inetaddress-key* write data in standard IPV4 or IPV6 address notation, depending on the length of source data in a flow. Aggregation schemes can include this *srcaddr* column in output by referencing the key's ID *srcaddr-key.*

The value *packet-count-value* is declared as an instance of the value builder sum-value, and references the field *pkts*. Because a *name* element is not specified, the column name in output defaults to *pkts*; all instances of *sum-value* write data as an integer value. Aggregation schemes can include this pkts column in output by referencing the value's ID *packet-count-value*.

Refer to the <*key-builders*> and <*value-builders*> elements in **nfc-config-predefined.xml** for the complete list of default keys and values. Keys and values for all V1-V8 fixed aggregation schemes are provided by default. The user can add additional keys and values or modify pre-defined keys and values as needed in the file **nfc-config.xml**.

Creating an Aggregation Scheme

An aggregation scheme is a declaration of the particular keys and values used to aggregate flow data and generate output. In CNS NetFlow Collection Engine, Release 3.x and 4.0, the fixed aggregation schemes for V1-V8 NDE were implemented in code and could not be changed. In CNS NetFlow Collection Engine Release 4.0, "cafeteria-style" aggregation was introduced where the user picks keys and values; however, there were limitations in the implementation (field mapping not supported for example), and there was an awkward separation between the two approaches (separate filter syntax for example).

In CNS NetFlow Collection Engine, Release 5.0, all aggregation is cafeteria-style. XML definitions are provided for all of the fixed aggregation schemes from CNS NetFlow Collection Engine, Release 3.x and 4.0 that can be modified as needed.

For example, the definition of the HostMatrix aggregation scheme is defined as follows:

```
<aggregation-scheme id="HostMatrix">
    <keys>
        <key id="srcaddr-key"/>
        <key id="dstaddr-key"/>
    </keys>
    <values>
        <value id="packet-count-value"/>
        <value id="byte-count-value"/>
        <value id="flow-count-value"/>
    </values>
</aggregation-scheme>
```

The ID attribute is the name of the aggregation scheme. Aggregators refer to the aggregation scheme by the ID, and output file headers contain the ID. Each key ID references a key defined in <*key-builders*>; the value ID references a value defined in <*value-builders*>.

As indicated previously, an aggregation scheme is independent of NDE version. The only requirement is that every field referenced by every key and value in the aggregation scheme must appear in flows that are aggregated. Otherwise, an error is reported as a log when the flow is processed.

For a complete list of default aggregation schemes, refer to the <*aggregation-schemes*> element in **nfc-config-predefined.xml**, where all fixed aggregation schemes from CNS NetFlow Collection Engine, Release 3.x and 4.0 are defined.

Creating an Aggregator

A thread in CNS NetFlow Collection Engine, Release 3.x and 4.0 is now called an aggregator. An aggregator tells CNS NetFlow Collection Engine how to aggregate (summarize) the traffic flows collected on the workstation on a port. Aggregators are defined in the <*aggregators*> element in **nfc-config.xml**.

Table 4-4 displays attributes that can be specified for an aggregator:

Table 4-4 Aggregator Attributes

Attribute	Description
id	Uniquely identifies the aggregator. This name appears in the output file path and header.
packet-log-enabled	Optional, default is false. If true, all packets from the port specified for this aggregator are logged for debugging purposes. Note that this can result in a significant reduction in performance.
device-name-format	Optional, default is address. If address, the router address in the output file path will be written as an ipaddress; if name, the router's DNS name is used instead.
is-output-sorted	Optional; default is false. If true, output records are sorted by key. Note that sorting output potentially reduces collector performance and throughput.

Table 4-5 displays elements that can be specified for an aggregator:

Table 4-5 Aggregator Elements

Element	Description
aggregation-scheme	The ID of the aggregation scheme to use.
period-minutes	Aggregation period.
port	Port number to listen on. An optional protocol attribute is included for future support of NDE transport protocols other than UDP.

Table 4-5 Aggregator Elements (continued)

Element	Description
state	Either active or inactive. If inactive, the aggregator is disabled.
filter	Optional. ID of the filter to apply to flows sent to this aggregator. Unlike CNS NetFlow Collection Engine, Release 3.x and 4.0, only one filter is specified because in Release 5.0 a filter can contain a complex expression.
writers	Contains one or more writer elements. In Release 5.0 this is ascii-writer or binary-writer. A writer contains additional elements as discussed below.
use-shortcut-address-as-source-ip	Optional, defaults to false. V7 and certain V8 NDE contain the field *router_sc* indicating the address of a router that was bypassed by the switch sending the flow. If true, for these versions of NDE the address of the switch sending Netflow data is replaced with the *router_sc* address in the flow.
output-v5-sampling- info	Optional, defaults to false. If true, optional sampling information in the NDE V5 header sent by certain types of routers is included in the CNS NetFlow Collection Engine output file header.

Table 4-6 displays attributes are specified for ascii-writer and binary-writer.

Note The use of binary-writer is discouraged, as support for binary output is being phased out starting with Release 5.0. If you want to reduce the size of output file, use compressed ASCII output instead.

Table 4-6 ascii-writer and binary-write Attributes

Attribute	Description
output-base-dir	Base directory where output files are written, such as **/opt/CSCOnfc/Data**.

Table 4-7 displays elements that can be specified for ascii-writer and binary-writer.

Table 4-7 ascii-writer and binary-writer Elements

Element	Description
use-compression	Optional, defaults to false. If true, output files are compressed in gzip format.
max-disk-usage- megabytes	Optional, defaults to 0.If a non-zero value, disk consumption for this aggregator is limited to the number of megabytes specified. Note that using this instead of the general cleanup capability can result in a reduction in performance.
filename-includes- date	Optional, defaults to false. If true, the output file suffix includes the date and time; if false, only the time is included.
output- postprocessors	List of programs or scripts to run for each output file when the file becomes available and before it is logged in the filesready file. It is critical to ensure that a postprocessing program or script should run quickly and never perform time-consuming tasks, since as long it is running, memory used for aggregation data cannot be released. A program that must perform a time-consuming task should instead listen for filesready file update events. The optional attribute *ignore-exit-status* attribute on each postprocessor indicates whether the program's exit status should be ignored. If false (default), a non-zero exit status results in an error log, and the filesready file is not updated with this filename. Output that the postprocessor writes to standard output and error is logged in **NFC_DIR/logs/nfc.log**.

The following is an example of an aggregator definition:

```
<aggregator id="HostMatrixExample">
    <aggregation-scheme id="HostMatrix"/>
    <period-minutes>5</period-minutes>
    <port protocol="udp">10000</port>
    <state>active</state>
    <writers>
        <ascii-writer>
            <use-compression>true</use-compression>
            <output-postprocessor>/var/tmp/copyNFCData.sh</output-postprocessor>
        </ascii-writer>
    </writers>
</aggregator>
```

This aggregator uses the HostMatrix aggregation scheme, collects data on UDP port 10000 and writes compressed ASCII output at five minute intervals. The user script **/var/tmp/copyNFCData.sh** is run for each output file.

Creating a Filter

Filter specification and behavior has changed significantly from CNS NetFlow Collection Engine releases prior to 5.0. In Release 5.0, a filter consists of an expression that evaluates to a boolean result. If the filter evaluates to true for a flow, the flow is aggregated; otherwise the flow is ignored. Logical operators are now supported. Nested filter expressions of arbitrary complexity are also supported.

A filter contains a top-level *expression* or *condition*. An *expression* contains two or more elements whose evaluation results are ANDed or ORed together; the result can then optionally be negated. An element is either a *condition* or another *expression*. Arbitrarily complex filter logic can therefore be defined by nesting expressions within the top-level expression.

A filter *condition* returns a boolean result by comparing the value returned by a key builder against one or more values or ranges of values specified in the condition definition. If the filter condition operation is *equals* and the set of values or ranges contains the value returned by the key builder, the condition evaluates to true. Similarly, the condition evaluates to true if the condition is *notequals* and the value doesn't appear in the set.

Consider the following example:

```
<filter id="FilterExample">
    <expression op="and">
        <expression op="or">
            <address-filter-condition key-builder="srcaddr-key" op="equals">
                <range>
                    <minimum>64.102.41.1</minimum>
                    <maximum>64.102.41.31</maximum>
                </range>
            </address-filter-condition>
            <address-filter-condition key-builder="dstaddr-key" op="equals">
                <range>
                    <minimum>64.102.41.1</minimum>
                    <maximum>64.102.41.31</maximum>
                </range>
            </address-filter-condition>
        </expression>
        <expression op="or">
            <integer-filter-condition key-builder="srcport-key" op="equals">
                <value>80</value>
            </integer-filter-condition>
            <integer-filter-condition key-builder="dstport-key" op="equals">
                <value>80</value>
            </integer-filter-condition>
        </expression>
    </expression>
</filter>
```

The filter in the example above permits flows only from source or destination addresses in the range 64.102.41.1 to 64.102.41.31 and where the source or destination port is 80. An aggregator referencing this filter would aggregate only flows matching these criteria.

The evaluation results of the two nested expressions are ANDed together by the top-level expression. The first nested expression has two conditions: the first checks for the source address with a specified range, and the second checks for the destination address within a specified range. The results of these two conditions are ORed together to yield the evaluation result of the first nested expression.

Note that for efficiency, CNS NetFlow Collection Engine performs *lazy* evaluations. For example, if two conditions are ORed together and the first condition evaluates to true, the second is not evaluated.

Multiple values and ranges can be specified within one condition if needed. Also, the additional expression operators *and* and *nor* are provided which negate the evaluation result returned for an expression. If in the example above nand was specified as the operator in the top-level expression instead of and, the filter would permit all flows except those matching the specified criteria.

Note that when evaluating one key builder result against several different target values, it is more efficient to specify multiple target values in a single condition than it is to specify several conditions.

The filter condition is specified as one of *address-filter-condition*, *integer-filter- condition*, or *string-filter-condition* depending on the data type of the key builder that is referenced. The special condition *nde-source-filter-condition* allows flows to be filtered based on router ipaddress or hostname.

Creating a Map

Map specification and behavior has changed significantly from releases prior to 5.0. In CNS NetFlow Collection Engine, Release 5.0, any key can be mapped including address keys; and any number of values or ranges of values can map to a label.

Two mapping key builder types are provided: *integer-range-map-key* and *address-range- map-key*. The integer map is specified for integer field types (e.g. source and destination ports, AS numbers, TOS); the address map is specified for address field types (e.g. source and destination addresses). XML syntax is the same for both, except that the integer map accepts integer values and ranges, whereas the address map accepts IPV4 and IPV6 values and ranges.

The following example illustrates how to specify a map:

```
<address-range-map-key id="MapExample">
    <field>srcaddr</field>
    <default-label>OTHER</default-label>
    <ranges>
        <range label="SITE1">
            <minimum>64.102.41.1</minimum>
            <maximum>64.102.41.254</maximum>
        </range>
        <range label="SITE2">
            <minimum>64.102.42.1</minimum>
            <maximum>64.102.42.254</maximum>
        </range>
        <range label="SITE3">
            <minimum>64.102.43.1</minimum>
            <maximum>64.102.43.254</maximum>
        </range>
    </ranges>
</address-range-map-key>
```

In this example, the value to be mapped is obtained from the source address field in a flow. Three address ranges are specified; if the address is in one of those ranges, it is mapped to the corresponding label. If the address is not within one of the ranges, it is mapped to the default label OTHER. If a default label had not been specified, the result of the mapping would be the string representation if the address itself.

Creating a Multi-Field Map

The multi-field map introduced in CNS NetFlow Collection Engine, Release 5.0 is a generic implementation of the protocol map in releases prior to 5.0. The old protocol map allowed combinations of the prot byte, source port, and destination port in a flow to be mapped to a label. However, with the multi-field map, any keys can be combined to produce a mapping. This can be particularly powerful because it allows mappings to be performed on the output of other key builders that transform flow data, for example, masked-inetaddress-key and bit-field-key.

The multi-field map is organized recursively, similar somewhat to a filter definition. The following illustrates the structure of a multi-field map:

```
refinement
    condition
        case-1
            labelopt
            refinementopt
        case-2
            labelopt
            refinementopt
        ...
    condition
        case-3
            labelopt
            refinementopt
        case-4
            labelopt
            refinementopt
        ...
    ...
```

Each condition references one key builder that extracts a value from a flow. Each case has one or more values or ranges of values. If there is a match and a label was specified, the value from the flow is mapped to that label. If a nested refinement was specified, the new set of conditions and therefore keys introduced in the nested refinement are evaluated. The search is continued until a match is found; if there is no match, a default label is used.

The protocol map from CNS NetFlow Collection Engine, Release 4.0 and earlier has been implemented as an instance of the multi-field map. The definition of protocol-map-key in nfc-config-predefined.xml is a good example of how to define a multi-field map. The definition is quite large, but a small portion appears below to help illustrate how the multi-field map works:

```
<multi-field-map-key id="protocol-map-key">
    <name>ProtocolExample</name>
    <default-label>OTHER</default-label>

    <refinement>
        <integer-map-condition key-builder="prot-key">
            <case>
                <value>1</value>
                <label>ICMP</label>
            </case>

            <case>
                <value>6</value>
                <label>TCP_OTHER</label>

                <refinement>
                    <integer-map-condition key-builder="srcport-key">
                        <case>
                            <range>
```

```
                                <minimum>20</minimum>
                                <maximum>21</maximum>
                            </range>
                            <label>TCP_FTP</label>
                        </case>
                    </integer-map-condition>
                </refinement>
            </case>
        </integer-map-condition>
    </refinement>
</multi-field-map-key>
```

This example defines a key whose column name in output is *ProtocolExample*. It has one top-level condition referencing the key builder *prot-key*. If the prot byte in a flow contains the value 1, a mapping is made to the label ICMP. If the prot byte contains the value 6, the mapped value depends on whether a match is found within the nested refinement for the nested condition that references *srcport-key*. If *srcport-key* is within the range 20 to 21, a mapping is made to TCP_FTP. Otherwise, the mapping is made to the label specified for the *prot-key* value 6, TCP_OTHER.

Depending on the type returned by the key builder that is chosen for a condition, one of *integer-map-condition*, *address-map-condition*, or *string-map-condition* must be specified as the conditions for a refinement. Because both conditions above reference builders that return an integer type, *integer-map-condition* is specified for both.

Note also that even if a value or range in a condition is matched, if no label is specified, then subsequent conditions will continue to be searched. This can be useful if a mapping is desired only when a nested refinement matches some value. To illustrate using the example above, suppose that TCP_OTHER was not specified as the label for the *prot-key* value 6. If a match did not occur within the nested refinement containing the *srcport-key* condition, the mapping would have reverted to the top-level default label OTHER.

Creating an Option Data Map

The Option Data Map caches option data entries that are used by option data key builders to map one or more fields in a data flow to a value from an option data flow. One or more fields in an option data flow are interpreted as keys that also must appear in the data flows for which mapping is to be performed. One and only one other field in the option data record is interpreted as a value that is associated with the unique combination of these key fields. An option data key builder performs a mapping to this value by doing a lookup in the Option Data Map.

In the same way an aggregation scheme defines the fields of interest for aggregating data flows, the option data map is configured with one or more option data map entries that define the key and value fields to be extracted from option data records. If an option data record contains all fields of interest, CNS NetFlow Collection Engine creates a map entry that can be retrieved by an option data key builder.

To create an option data map entry, do the following:

Step 1 Locate the field definitions for the fields of interest that are correlated between data flows and option data flows. For example, an option data flow is sent with mapping information for interface indexes and interface names. Locate the field definition for these fields in the `<fields>` element in **nfc-config-predefined.xml**:

```
<field id="10" name="INPUT_SNMP" type="integer"/>
<field id="14" name="OUTPUT_SNMP" type="integer"/>
<field id="82" name="IF_NAME" type="utf8-string"/>
```

Step 2 Use existing key builders or create new key builders that reference these fields to obtain the data from both option and traffic data flows:

```
<integer-key id="input-if-index-key">
    <field>INPUT_SNMP</field>
        </integer-key>
            <integer-key id="output-if-index-key">
            <field>OUTPUT_SNMP</field>
        </integer-key >
        <string-key id="if-name-key">
    <field>IF_NAME</field>
</string-key>
```

Step 3 Create an `option-data-map-entry` element that references these fields within an `option-data-map` in **nfc-config.xml** so that matching entries are created and cached in the option data map.

The entry in the example below causes option data records with the `INPUT_SNMP` and `IF_NAME` fields to be placed in the option data map with the entry ID **if-name-map-entry**. Each entry contains one key: the integer returned by the **input-if-index-key** key builder; and the interface name string value returned by the **if-name-key** key builder defined above. Note that in this example, the field type of the interface index in option data flows is the input interface index only. The router only sends the input interface in the option data record because the interface name is the same regardless of whether the index is an input or output interface.

```
<option-data-map>
    <option-data-map-entry id="if-name-map-entry">
        <keys>
            <key id="input-if-index-key"/>
        </keys>
        <value id="if-name-key"/>
    </option-data-map-entry>
</option-data-map>
```

Step 4 Create one or more option data key builders that reference this option data map entry to map from fields in traffic flows to values in the option data cache. The **input-if-name-map-key** and **output-if-name-map-key** key builders in the example below map the input interface index and output interface index in data flows to the interface name value in an **if-name-map-entry** option data map entry.

```
<option-data-key id="input-if-name-map-key">
    <name>input_interface_name</name>
        <option-data-map-entry>if-name-map-entry</option-data-map-entry>
            <keys>
                <key id="input-if-index-key"/>
            </keys>
        </option-data-key>
        <option-data-key id="output-if-name-map-key">
    <name>output_interface_name</name>
        <option-data-map-entry>if-name-map-entry</option-data-map-entry>
            <keys>
                <key id="output-if-index-key"/>
            </keys>
        </option-data-key>
```

If no map entry exists for a particular key, the result in the CNS NetFlow Collection Engine output is an empty string, that is the column value in output is empty. Applications that process CNS NetFlow Collection Engine output files should expect this, because right after CNS NetFlow Collection Engine starts, there is always a window of time so that data flows arrive prior to option data flows.

Note In CNS NetFlow Collection Engine, 5.0.2 option data map entries cannot be created with the web-based user interface; they must be created by editing the configuration XML. For additional information about option data key builders see the "option-data-key" section on page 4-11.

Creating Source Groups

Normally, flows from different devices are aggregated separately, and separate output files are written for each device. However, flows from more than one device can be aggregated and output together by specifying an NDE source group that lists each device in the group.This is the ROUTER_GROUPNAME feature in releases prior to CNS NetFlow Collection Engine, Release 5.0.

The following is an example of how to specify an NDE source group:

```
<nde-source-groups>
    <group id="group1">
        <nde-source id="64.102.41.1"/>
        <nde-source id="64.102.41.2"/>
        <nde-source id="64.102.41.3"/>
    </group>
</nde-source-groups>
```

Groups are added to the nde-source-groups element in **nfc-config.xml**. NDE that is received from each of the three devices in group1 above is aggregated and output together under the device name group1.

Creating Access Lists

A global access list is provided for allowing or denying packets from a specific set of devices. This is the ACCEPT_PACKETS_FROM feature in releases prior to CNS NetFlow Collection Engine, Release 5.0, except that this can also be configured to reject packets only from devices in the set.

The following is an example of how to specify an access list:

```
<nde-source-access-list action="permit">
    <nde-sources>
        <nde-source id="64.102.41.10"/>
    </nde-sources>

    <nde-source-groups>
        <group id="group1"/>
    </nde-source-groups>
</nde-source-access-list>
```

In this example, packets are accepted only from the device 64.102.41.10, and from any devices listed in *group1*. If the *action* attribute of *nde-source-access-list* had been *deny*, packets would have been accepted from all sources except those listed.

Global Settings

Table 4-8 displays the global settings found in **nfc-config-predefined.xml**. In all pathnames in Global Settings, the top-level CNS NetFlow Collection engine directory can be indicated as **${NFC_DIR}**. These settings and can be overridden in **nfc-config.xml**.

Table 4-8 Global Settings

Setting	Description
num-packet-pool- entries	Internal packet buffer size setting. Internal use only.
max-nde-packet-size	Maximum size of an NDE packet. Internal use only.
cleanup-interval	Time duration in hours between each cleanup job.
cleanup-job	Location of the executable that is invoked at the end of each *cleanup-interval*.
filesready-file-dir	Absolute path of directory where filesready files are written.
packet-log	Settings for packet logging. The *base-dir* attribute is the absolute path of the directory where packet log files are written. The *use-compression* attribute indicates whether packet log files are compressed.
output-field-delimiter	Separator character for fields in output, either a comma or vertical bar.
start-output-at-top-of-the-hour	If *true*, the time that an output file is first written after the collector restarts is calculated from the top of the hour. If *false*, output periods are relative to the collector start time.
output-format	One of *csv-only*, *mixed*, or *xml-only*. If csv-only, the CNS NetFlow Collection Engine output file header is compatible with CNS NetFlow Collection Engine, Release 3.x/4.0. If *mixed*, the output file header is XML that contains additional information about each field, although data records are still written as comma- or bar-separated fields. If *xml-only*, the entire output file is written as XML. Note that for output files to be compatible with the CNS NetFlow Collection Engine, 5.0 reporting feature, *output-format* must be *mixed*.

Setting the Time Zone

The date and time used in naming the data file directory structure, names of data files, headers in data files, and messages in the log files can be changed to be relative to a time zone different than the local time zone. Prior releases of CNS NetFlow Collection Engine used a global setting for this. However, CNS NetFlow Collection Engine, Release 5.0 uses the standard UNIX environment setting TZ to accomplish this.

To change the time zone used by CNS NetFlow Collection Engine from the local time zone to GMT, add the following at the beginning of the startup script **/opt/CSCOnfc/bin/nfcollector**:

TZ=GMT

export TZ

Tuning Memory Usage

CNS NetFlow Collection Engine memory requirements are determined by the following factors:

- Number of source devices
- Amount and "uniqueness" of Netflow data within an aggregation period
- Number of aggregators
- Number and type of keys and values in each aggregation scheme
- Overhead, for example pre-allocated buffers, and program size

In previous releases, CNS NetFlow Collection Engine programs were compiled executables that in a poorly-scaled deployment would consume memory until all system memory was exhausted. However, because CNS NetFlow Collection Engine 5.0 is built primarily with Java technology, an absolute maximum on how much memory the Java Virtual Machine can allocate is specified on the JVM's command line.

If insufficient memory has been specified as the upper bound, the JVM reports an out-of-memory condition, and CNS NetFlow Collection Engine attempts to output logs indicating what has happened. If logs are reported in any of the log files under **$NFC_DIR/logs** indicating that memory has been exhausted, the upper bound can be increased if additional system memory is available.

Memory settings for all CNS NetFlow Collection Engine processes are consolidated in the file **$NFC_DIR/config/nfcmem**. For each process, two settings are defined: the initial and the maximum memory that can be used by the process. For example, one of the largest potential consumers of memory is the core aggregation process. Default settings for the initial and maximum amounts of memory this process can allocate are defined in nfcmem as follows:

```
COLLECTOR_MEM_INI=-Xms256M
COLLECTOR_MEM_MAX=-Xmx768M
```

-Xms256M specifies that the starting size is 256 megabytes; **-Xmx768M** specifies that up to 768 megabytes can be allocated as needed to aggregate Netflow data. These values were chosen as initial defaults for a 1 gigabyte machine dedicated to running CNS NetFlow Collection Engine; other CNS NetFlow Collection Engine processes will consume additional memory.

For example, generating reports might require more memory than the default for the web process. To set the initial and maximum memory sizes for the web process to 128MB and 256MB respectively, set **WEB_MEM_INI** to **-Xms128M** and **WEB_MEM_MAX** to -**Xms256M** in file

$NFC_DIR/config/nfcmem. The settings 128MB and 256MB are suggestions and might not be adequate. If memory-related errors continue to occur when generating reports, higher values may be necessary.

The following table lists each memory setting in **nfcmem** and the corresponding process; the associated process watcher subsystem is noted in parentheses if applicable:

Table 4-9 nfcmem settings

Memory Setting	Corresponding Process
COLLECTOR_MEM_INI, COLLECTOR_MEM_MAX	Core collection and aggregation (nfc)
RE_MEM_INI, RE_MEM_MAX	Reporting engine (re)
WEB_MEM_INI, WEB_MEM_MAX	Web server (web)
BGP_MEM_INI, BGP_MEM_MAX	CNS NetFlow Collection Engine BGP peer
CNSXML_MEM_INI, CNSXML_MEM_MAX	CNS-XML interface (nfcxml)
GEN_REPORT_MEM_MIN, GEN_REPORT_MEM_MAX	Command-line reporting tool

The **nfcmem** setting **UDP_READER_POOL_ENTRIES s**pecifies the number of entries in an internal packet buffer. The actual memory consumed by the buffer is this value times the size of each buffer entry, which is determined by the global setting **max-nde-packet-size**. Although the default value of **UDP_READER_POOL_ENTRIES** is not typically updated, for low peak throughput rates of NetFlow data on a low-end system, reducing this value can conserve memory, but with added risk of packets being dropped.

Starting in Release 5.0.3, a memory monitor feature can be enabled that proactively frees system memory by dropping NetFlow packets and aggregated output when too much NetFlow data is sent to a collector for an extended period. This feature is disabled by default. Because a large amount of data is potentially discarded and because the feature activates at less than full memory occupancy, most users will probably choose not to enable this feature.

To enable the memory monitor feature, uncomment or add a memory-monitor configuration element in the file **/opt/CSCOnfc/config/nfc-config.xml** immediately following the disk-usage-monitor configuration element as follows:

```
<memory-monitor
mem-check-period-millis="2000"
suspend-packet-processing-mem-usage-percent="85"
drop-queued-output-jobs-mem-usage-percent="95"
resume-normal-operation-mem-usage-percent="75"
/>
```

In this example, the memory monitor polls memory usage every 2 seconds (2000 milliseconds). When memory usage reaches 85% of the limit configured for **COLLECTOR_MEM_MAX** in the file **nfcmem**, NetFlow packets are discarded until memory usage drops below 75%. When memory usage reaches 95% of the limit, aggregated data queued for output is discarded. Discarded packets and queued output data are not recoverable. Logs are generated when packets and queued output data are discarded, and again when normal operation resumes.

Managing Disk Space

Depending on the volume of flow data being exported from the export devices and the CNS NetFlow Collection Engine thread attribute settings you use, CNS NetFlow Collection Engine can consume large amounts of disk space in a short period. CNS NetFlow Collection Engine provides several features that can help you manage your disk space usage:

- Filters
- Aggregation
- Data file and disk space options

Filters

As described earlier, a filter can help you discard any flow data that is not of interest to you. By using filters to ensure that you are storing only data of interest, you can potentially reduce the amount of disk space used by CNS NetFlow Collection Engine.

Aggregation

Aggregation schemes are used to define how you want CNS NetFlow Collection Engine to summarize the flow data being exported from your export devices. By using only those aggregation schemes required for your application and, when possible, by selecting the aggregation schemes that generate the least amount of data on disk, you can reduce the amount of disk space used by CNS NetFlow Collection Engine. For example, using the **HostMatrix** aggregation scheme results in less disk space usage than using the **DetailHostMatrix** scheme. Of course, the aggregation schemes you use are determined primarily by the data you are interested in and how you want to summarize that data. It is important to realize, however, that the different aggregation schemes can greatly affect the amount of disk space used by CNS NetFlow Collection Engine.

You can estimate the amount of UDP traffic that an export device generates when NetFlow data export is enabled. To do this you must understand the characteristics of the traffic in your network, including the average packets per second of switching throughput and the average number of packets per flow.

For example, if the average throughput on a NetFlow enabled export device is 150 packets per second and the average number of packets per flow is 100, you could have approximately 1500 flow records per second (150 x 100) to be exported by the export device. If NetFlow data export format Version 5 datagrams are used, you should expect approximately 50 NetFlow export datagrams per second (1500 flows/30 per export datagram) or 45 KB per second (30 x 1500 bytes per datagram) from the export device.

Data File and Disk Space Options

Optional parameters are available to limit disk space and improve system performance at the same time. These parameters are documented in the "Creating an Aggregator" section on page 4-16 and "Global Settings" section on page 4-25. You can limit disk space and improve system performance by doing any of the following:

- Apply **gzip** compression to data files.
- Specify the amount of disk space used by data files before they are removed using the **max-disk-usage-megabytes** parameter.

Note Specifying max-disk-usage for an aggregator instead of using the periodic cleanup capability can result in a significant reduction in performance.

- Specify the amount of time between each data file **cleanup-job** using the **cleanup-interval** parameter.
- Specify the program or script to run at the end of the **cleanup-interval** parameter to flush data files using the cleanup-job parameter.

Monitoring Disk Usage

CNS NetFlow Collection Engine includes a disk usage monitor utility that writes a warning log to the log file **$NFC_DIR/logs/nfc.log** when disk usage reaches or exceeds a configurable limit. Once the warning log is written, an informational log is written when disk usage drops below a separate configurable limit. More that one filesystem can be monitored if needed.

To monitor disk usage, create a `disk-usage-monitor` element in XML configuration. For each filesystem to be monitored, include a monitor element.

For example, with the following configuration the filesystem containing the **$NFC_DIR/Data** directory is checked at one minute intervals; a warning log is issued when usage reaches 90% and an informational log is subsequently issued when usage drops below 80%:

```
<disk-usage-monitor>
<monitor filesystem="${NFC_DIR}/Data" interval="1" warning-threshold="90"
clear-threshold="80"/>
</disk-usage-monitor>
```

NetFlow Collection Engine Logger Configuration

The following sections describe the NetFlow Collection Engine logger configuration feature available in Release 5.0.3.

Rolling File Option for NFC Logs Files

Note The NetFlow Collection Engine GUI cannot display rollover log files. Only active log files are displayed in the web browser. The **nfcudprd.log** file on Solaris and HP-UX systems is not affected by these log4j configuration changes.

In order to create rolling log files, modify your ***xxx*-log4j. properties** file to look like the following:

```
log4j.rootLogger=OFF
log4j.loggerFactory=com.cisco.nfc.collector.logging.NFCLoggerFactory

log4j.logger.com.cisco.nfc=INFO, nfcLog
log4j.appender.nfcLog=org.apache.log4j.RollingFileAppender
log4j.appender.nfcLog.File=${NFC_DIR}/logs/nfc${NFC_LOGFILE}.log
log4j.appender.nfcLog.MaxFileSize=100KB
log4j.appender.nfcLog.MaxBackupIndex=3
log4j.appender.nfcLog.layout=org.apache.log4j.PatternLayout
```

```
log4j.appender.nfcLog.layout.ConversionPattern=[%d{yyyy-MM-dd HH:mm:ss zz}] %p %c - %m%n
```

Note This configuration will create a max of 3 rolling files each of size 100KB.

Configurable options for rolling files include:

- **MaxFileSize**: size of each rolling file (you can specify the size in KB or MB)
- **MaxBackupIndex**: max number of rolling files.

When rollover occurs, the old version of **nfc.log** is automatically moved to **nfc.log.1**.

Daily Rolling File Option for NFC Log Files

Using the **DailyRollingFileAppender** option, you can specify how often files are rolled over. The rolling schedule is specified by the **DatePattern** option.

The example below shows the **DailyRollingFileAppender** option for the **nfc-log4j.properties** file. The following configuration will rollover a file every day at midnight.

Note The rollover file frequency is only determined when there is a message to be logged. If there are no log messages being logged then the rollover will not occur until another log message arrives in order to avoid empty log files. As a result, log files for certain periods might be missing due to the logger being inactive during those periods.

```
log4j.rootLogger=OFF
log4j.loggerFactory=com.cisco.nfc.collector.logging.NFCLoggerFactory

log4j.logger.com.cisco.nfc=INFO, nfcLog
log4j.appender.nfcLog=org.apache.log4j.DailyRollingFileAppender
log4j.appender.nfcLog.File=${NFC_DIR}/logs/nfc${NFC_LOGFILE}.log
log4j.appender.nfcLog.DatePattern='.'yyyy-MM-dd
log4j.appender.nfcLog.layout=org.apache.log4j.PatternLayout
log4j.appender.nfcLog.layout.ConversionPattern=[%d{yyyy-MM-dd HH:mm:ss zz}] %p %c - %m%n
```

Note Be aware that you are responsible for file management tasks when using this option. Delete the rollover files at a regular interval to avoid disk full situation.

You can specify monthly, weekly, half-daily, daily, hourly, or minutely rollover schedules with the **DailyRollingFileAppender** option. Refer to Table 10.

Table 10 Rollover Schedule Elements

Date Pattern	Rollover Schedule	Example
'.'yyyy-MM	Beginning of each month.	At midnight of May 31st, 2005 **nfc.log** will be copied to **nfc.log.2005-05**. Logging for the month of June will be output to **nfc.log** until it is rolled over the next month.
.'yyyy-ww	First day of each week. The first day of the week depends on the locale.	Assuming the first day of the week is Sunday, on Saturday midnight, June 9th 2005, the file **nfc.log** will be copied to **nfc.log.2005-23**. Logging for the 24th week of 2005 will be output to **nfc.log** until it is rolled over the next week.
''.'yyyy-MMdd	Midnight each day.	At midnight, on March 8th, 2005, **nfc.log** will be copied to **nfc.log.2005-03-08**. Logging for the 9th day of March will be output to **nfc.log** until it is rolled over the next day.
'.'yyyy-MMdd- a	Midnight and midday of each day.	At noon, on March 9th, 2005, **nfc.log** will be copied to **nfc.log.2005-03-09-AM**. Logging for the afternoon of the 9th will be output to **nfc.log** until it is rolled over at midnight.
`'.'yyyy-MM-dd-HH`	Top of every hour.	At approximately 11:00.000 o'clock on March 9th, 2005, **nfc.log** will be copied to **nfc.log.2005-03-09-10**. Logging for the 11th hour of the 9th of March will be output to **nfc.log** until it is rolled over at the beginning of the next hour.
.'yyyy-MMdd- HH-mm	Beginning of every minute.	At approximately 11:23,000, on March 9th, 2005, **nfc.log** will be copied to **nfc.log.2005-03-09-10-22**. Logging for the minute of 11:23 (9th of March) will be output to **nfc.log** until it is rolled over the next minute.

Configuring the Logger from the Command Line

You can use the **logconfig.sh** script located in the **{NFC_HOME_DIR}/tools** directory to configure all the NFC log files. By default, the rollover file feature is disabled.

Note The **nfcudprd.log** file on Solaris and HP-UX systems is not affected by these **log4j** configuration changes.

To configure the logger, do one of the following:

Option 1: Enter the following command to configure the logger to enable rollover based on log file size and to limit the number of rollover files:

```
./logconfig.sh -filesize sizes [KB|MB] -maxfiles num
./logconfig.sh -filesize 100KB -maxfiles 3
```

Option 2: Enter the following command to configure the logger to enable rollover based on a specified rollover period:

```
./logconfig.sh -period hourly| twicedaily|daily|weekly|monthy
./logconfig.sh -period daily
```

Option 3: Enter the following command to configure the logger to disable the rollover file feature.

```
./logconfig.sh –default
```

Note T he **logconfig.sh** script overwrites any changes that may have been made to the log configuration files. If you have modified the properties files, try to configure the logger manually following the instructions in the two previous subsections.

CNS NetFlow Collection Engine Advanced Features

This chapter contains information on advanced features contained in CNS NetFlow Collection Engine, Release 5.0.

This chapter includes the following sections:

- Process Watcher, page 5-1
- Event Service, page 5-3
- Report Generator, page 5-4
- BGP Peer, page 5-8
- Interface Name Support, page 5-10

Process Watcher

The Process Watcher provides high availability to the CNS NetFlow Collection Engine system. It is responsible for starting, stopping, and restarting CNS NetFlow Collection Engine processes. It monitors CNS NetFlow Collection Engine processes and attempts to restart a process, up to the configured number of restarts, if the process dies unexpectedly with a non-zero return status.

The nfcollector script starts the Process Watcher the first time a managed process is started. Once started, the Process Watcher remains running until **nfcollector shutdown** or **nfcollector clean** is executed.

Table 5-1 displays the nfcollector command line arguments.

Table 5-1 nfcollector Command Line Arguments

Argument	Description
start all	Starts all managed processes marked for autostart.
stop all	Stops all managed processes marked for autostart.
start <managed process id>	Starts the managed process with the corresponding id attribute. The Process Watcher, and consequently all autostart processes, are started if necessary.
stop <managed process id>	Stops the managed process with the corresponding id attribute.

Table 5-1 nfcollector Command Line Arguments (continued)

Argument	Description
status	Lists all managed processes, whether or not they are running, and the process ID if one is stored in a file designated by the <pid-file> setting of a managed process.
shutdown	Gracefully stops all managed processes including the Process Watcher.
clean	Forcefully stops all managed processes including the Process Watcher.
show-tech	Gathers debugging information into a log file. As long as the user has write permission to **NFC_DIR/logs** the file generated by **show-tech** will be **NFC_DIR/logs/show-tech.log**, otherwise the data is written to **/tmp/show-tech.log**.

The Process Watcher can be started at system boot time with the **csco_nfcd** script. If you configure CNS NetFlow Collection Engine during installation to start at boot time, then system-dependent steps are taken to invoke the **csco_nfcd** script during system initialization and shutdown.

This script invokes **nfcollector start all** at system initialization and **nfcollector shutdown** at shutdown.

Configuration

The configuration for the Process Watcher is stored in **NFC_DIR/config/nfcpw.xml**. XML element text values in this file may contain **${NFC_DIR}** and the Process Watcher will substitute that string with the **NFC_DIR** environment variable. Each managed process must have an id attribute and the child elements as described Table 5-2.

Table 5-2 Configuration Elements

Child Element	Description	Required
commandline	What will be executed when the Process Watcher attempts to start the managed process.	Yes
stop-commandline	If defined, the Process Watcher will execute this command line when it attempts to stop the managed process. If omitted, the Process Watcher uses a system-dependent means of stopping the process.	No
autostart	A value of true tells the Process Watcher to automatically start the managed process when the Process Watcher starts or when the “start all” arguments are used with the nfcollector script. Default is false.	No
restart	A value of true tells the Process Watcher to monitor and restart the managed process if it dies unexpectedly with a non-zero exit status. Default is false.	No

Table 5-2 Configuration Elements (continued)

Child Element	Description	Required
restart-attempts	The number of times the Process Watcher should attempt to restart a process.	No
pid-file	If the managed process generates a file containing the process id of the managed process then the Process Watcher can include that information in its status output. Set this value to the path of that file.	No

For example, the CNS NetFlow Collection Engine collection process is configured with the following XML:

```
<managed-process id="collection">
  <commandline>${NFC_DIR}/bin/startnfc.sh</commandline>
  <autostart>true</autostart>
  <restart>true</restart>
  <restart-attempts>3</restart-attempts>
  <pid-file>${NFC_DIR}/logs/nfc.pid</pid-file>
</managed-process>
```

Event Service

When certain situations arise, CNS NetFlow Collection Engine will generate an event to notify interested parties. Such situations include when the collector is started and when it is stopped. The event service delivers these events to interested parties by way of event transports. Currently the only transport available is one for sending events out on the CNS Integration Bus.

Configuration

By default, the event service is configured to send events out on the CNS Integration Bus using a subject of **cisco.mgmt.nfc.event**. You can change this behavior by redefining the event service in **NFC_DIR/config/nfc-config.xml**. Add the following XML to the <*nfc*> element (see **NFC_DIR/config/nfc-config.xsd** for placement) to customize the subject for events on the CNS Integration Bus:

```
<event-service>
  <cns-event-bus-transport class="com.cisco.nfc.collector.event.CNSEventBusTransport">
    <subject>mySubject</subject>
  </cns-event-bus-transport>
</event-service>
```

You can optionally configure values for service, network, and daemon with XML elements of the same name as necessary for working with the CNS Integration Bus.

Message Format

The details of how to use the CNS Integration Bus are outside the scope of this document.

Event messages received on the CNS Integration Bus include the following information.

Field or Attribute	Description
DATA	The XML representation of the event

Sample Output

Here is a sample of an event generated when the collector is started:

```
<?xml version="1.0" encoding="UTF-8"?>
<nfc-event version="2.0">
    <start-collector-event>
        <collector id="64.102.41.53"/>
        <timestamp>11/6/03 1:13 PM</timestamp>
    </start-collector-event>
</nfc-event>
```

Here is a sample of an event generated when the collector is stopped:

```
<?xml version="1.0" encoding="UTF-8"?>
<nfc-event version="2.0">
    <stop-collector-event>
        <collector id="64.102.41.53"/>
        <timestamp>11/6/03 3:33 PM</timestamp>
    </stop-collector-event>
</nfc-event>
```

Report Generator

Report Generator is a separate process that produces reports based on CNS NetFlow Collection Engine data files. The configuration of the Report Generator allows generation of hourly and daily reports by performing further aggregation of the records in CNS NetFlow Collection Engine data files specified.

You can configure the following:

- The maximum number of records to include in the report and include the “others” category for the remaining records.
- Whether the reports combine output from different export devices or keep output separated per device.
- The automate clean up or removal, of old reports.

The default Report Generator configuration, **/opt/CSCOnfc/config/nfcre.xml.** The following is a sample configuration and its contents:

```
<nfc-reporting-config>
<report-config id="DailyFlowStats" frequency="daily" retain="7">
<data-dir>/opt/CSCOnfc/Data</data-dir>
<combine-device-output>true</combine-device-output>
<maximum-records order="descending" field="octets">10</maximum-records>
<output-path>/opt/CSCOnfc/Reports</output-path>
```

```
<thread id="DetailHostMatrixTest"/>
<keys>
<key field="srcaddr"/>
<key field="dstaddr"/>
<key field="protocol"/>
</keys>
<values>
<value>pkts</value>
<value>octets</value>
<value>flows</value>
<value>starttime</value>
</values>
</report-config>
</nfc-reporting-config>
```

A description of important attributes and elements in the XML configuration is given in Table 5-3:

Table 5-3 XML Configuration Elements

Attribute/Element	Description
id	Report identification.
frequency	Reporting frequency indicating how often the reports are generated. Default: **daily**.
retain	Number of days that a report will be retained. Default: **7**, meaning reports older than 7 days will be purged. Purging occurs at midnight everyday.
include-others	A boolean attribute to specify whether to include the record with key value of **Others**. This record is a result of aggregating all flows (during the reporting period) that are not counted in other records of the current report. Default: **true**.
data-dir	Top data directory under which CNS NetFlow Collection Engine data files are stored. Default: **/opt/CSCOnfc/Data**.
combine-device-output	Whether the reports combine output from different devices or keep output separated on per device basis. Default: **true**.
maximum-records	Number of records to include in the report; remaining records are aggregated into a special record named <*others*>. Default: all records are included.
order	Order of the records. Default: **descending**.
field (in maximum- records)	The field based on which the records are ordered. Any numeric value field can be used here.
id (in thread)	The name of aggregator based on whose output reports are generated.
output-path	Directory where reports are stored. Default: **/opt/CSCOnfc/Reports**.
field (in key)	Name of the key field.
value	Name of value field.

Note Rate values are not supported in either custom or scheduled reports in this release.

With this description you can see that the default Report Generator configuration specifies to report daily on the top 10 talkers based on output of aggregator **DetailHostMatrixTest**. The key combination consists of three keys: **srcaddr**, **dstaddr** and **protocol**.

Report Generator is not configured for autostart. Use the following command to start Report Generator:

/opt/CSCOnfc/bin/nfcollector start re

Note Report Generator is not marked for autostart so **nfcollector start all** will not start it. To launch Report Generator, you must use the **/opt/CSCOnfc/bin/nfcollector start re** command.

The reports can be viewed in the web UI. See the "Scheduled Reports" section on page 2-30 for information.

A text version of sample daily report, **DailyFlowStats_2003_11_04.0000**, looks like the following:

```
<report id="DailyFlowStats">
  <start-time>03-Nov-2003 00:00:00</start-time>
  <end-time>04-Nov-2003 00:00:00</end-time>
  <keys>
    <key>srcaddr</key>
    <key>dstaddr</key>
    <key>protocol</key>
  </keys>
  <values>
    <value>pkts</value>
    <value>octets</value>
    <value>flows</value>
    <value>starttime</value>
  </values>
  <exporting-device id="*">
    <records>
      <record>
        <srcaddr>0.0.0.26</srcaddr>
        <dstaddr>0.0.0.26</dstaddr>
        <protocol>ICMP</protocol>
        <pkts>90</pkts>
        <octets>90</octets>
        <flows>90</flows>
        <starttime>1067847919</starttime>
      </record>
      <record>
        <srcaddr>0.0.0.13</srcaddr>
        <dstaddr>0.0.0.13</dstaddr>
        <protocol>ICMP</protocol>
        <pkts>89</pkts>
        <octets>89</octets>
        <flows>89</flows>
        <starttime>1067847919</starttime>
      </record>
      <record>
        <srcaddr>0.0.0.1</srcaddr>
        <dstaddr>0.0.0.1</dstaddr>
        <protocol>ICMP</protocol>
        <pkts>29</pkts>
        <octets>29</octets>
        <flows>29</flows>
```

```
        <starttime>1067847919</starttime>
      </record>
      <record>
        <srcaddr>0.0.0.8</srcaddr>
        <dstaddr>0.0.0.8</dstaddr>
        <protocol>ICMP</protocol>
        <pkts>9</pkts>
        <octets>9</octets>
        <flows>9</flows>
        <starttime>1067847919</starttime>
      </record>
      …
      <record>
        <srcaddr>0.0.0.27</srcaddr>
        <dstaddr>0.0.0.27</dstaddr>
        <protocol>ICMP</protocol>
        <pkts>9</pkts>
        <octets>9</octets>
        <flows>9</flows>
        <starttime>1067847919</starttime>
      </record>
      <record>
        <others/>
        <pkts>954</pkts>
        <octets>954</octets>
        <flows>954</flows>
        <starttime>1067847919</starttime>
      </record>
    </records>
  </exporting-device>
 </report>
```

Report Generator can also be configured to produce reports that only contain records meeting certain criteria. Such criteria can be specified by providing a fixed value to one or more <*key*> elements in report configuration. For example:

```
<nfc-reporting-config>
…
<keys>
<key field="srcaddr">1.1.1.1</key>
<key field="dstaddr">2.2.2.2</key>
<key field="protocol"/>
</keys>
<values>
<value>pkts</value>
<value>octets</value>
<value>flows</value>
<value>starttime</value>
</values>
…
</nfc-reporting-config>
```

In this example, only those records with **srcaddr** equal to **1.1.1.1** and **dstaddr** equal to **2.2.2.2** will be aggregated and included in the reports.

Currently Report Generator can only report on ASCII output with XML self- describing headers.

BGP Peer

CNS NetFlow Collection Engine, Release 5.0 includes a Border Gateway Protocol (BGP) peer for supplementing CNS NetFlow Collection Engine output with BGP attributes. The peer is a passive peer, it does not advertise any prefixes or send any update messages. It establishes BGP sessions with all configured peers and stores BGP attributes for IP prefixes on a per-peer basis. Key builders are provided to query the CNS NetFlow Collection Engine BGP peer for BGP attributes using addresses from the NetFlow record. See the "bgp-attr-post-agg-key, bgp-attr-key" section on page 4-6 and the "bgp-complete-as-path-post-agg-key, bgp-complete-as-path-key" section on page 4-7 for additional information. When looking up attributes, the CNS NetFlow Collection Engine BGP peer does a longest match on the destination address against advertised prefixes for the BGP peer.

Starting and Stopping the BGP Peer

To start the CNS NetFlow Collection Engine BGP peer, run **/opt/CSCOnfc/bin/startbgp.sh &**. Be sure that the configuration in **NFC_DIR/config/nfcbgp.xml** is set up correctly to establish BGP sessions with devices exporting NetFlow data to the collector. See the "Configuration" section on page 5-8.

Note If running the CNS NetFlow Collection Engine BGP peer on the default port of 179 you must be root to start the process.

The **csco_nfcd** script located in the system-dependent startup script directory (for example, **/etc/rc.d**) can start and stop the BGP peer during system initialization and shutdown respectively. If you want to have the BGP peer automatically started and stopped with the system, add the lines in bold to the **csco_nfcd** script:

Start:

```
su $NFC_USER -c "$BINDIR/nfcollector start all"
$BINDIR/startbgp.sh &
;;
```

Stop:

```
su $NFC_USER -c "$BINDIR/nfcollector shutdown"
$BINDIR/stopbgp.sh &
;;
```

Because the root account runs these scripts, the BGP peer will be run as root when started by the **csco_nfcd** script.

Configuration

You can configure the BGP settings of the CNS NetFlow Collection Engine BGP peer in **NFC_DIR/config/nfcbgp.xml** with the elements shown in Table 5-4. These XML elements are children of a <*bgp-peer*> root element.

Table 5-4 BGP Configuration Elements

Element	Description	Required
port	Port on which the CNS NetFlow Collection Engine BGP peer listens for BGP messages. Default: 179 (standard BGP port value).	No
command-port	Port on which the CNS NetFlow Collection Engine BGP peer listens for queries for BGP attributes. Default: 7777.	No
bgp-id	BGP identifier for the CNS NetFlow Collection Engine BGP peer. Default: IP address of collector machine.	No
path-separator	Separator between paths in the result of a complete AS path lookup.	No
holdtime	Maximum amount of time in seconds allowed between BGP messages before session is timed out. The CNS NetFlow Collection Engine BGP peer sends a KEEPALIVE message every holdtime -1 seconds to maintain the session. Default: 30 seconds.	No

For each BGP peer, configure a BGP session with a <*bgp-session*> element in **NFC_DIR/config/nfcbgp.xml** with the child elements shown in Table 5-5. The <*bgp-session*> elements are also children of the root <*bgp-peer*> element.

Table 5-5 Additional BGP Configuration Elements

Element	Description	Required
peer-address	The IP address or hostname of the BGP peer.	Yes
peer-port	The port on which the BGP peer is listening for BGP messages. Default: 179 (standard BGP port value).	No
peer-asn	Autonomous System Number of the BGP peer.	Yes
alias	If the BGP peer is a device that is grouped by CNS NetFlow Collection Engine into a NDE source group, use this element to identify the group name used by CNS NetFlow Collection Engine.	No
nde-source	Specify an *nde-source* element for the IP address of each interface that exports NetFlow data. If **ip flow-export source *interface*** is configured on the device to correspond with the BGP peer address, *nde-source* should not be specified here. **Note:** In Release 5.0.3, *nde-source* elements cannot be configured through the web-based user interface.	No

For example, the following configuration establishes a BGP session with the router at 10.0.0.1, which has an ASN of 501.

```
<bgp-peer>
    <bgp-session>
        <peer-address>10.0.0.1</peer-address>
        <peer-asn>501</peer-asn>
    </bgp-session>
</bgp-peer>
```

Interface Name Support

CNS NetFlow Collection Engine, Release 5.0 includes the option to map interface indices to interface names and allow the interface name to be an aggregation key. CNS NetFlow Collection Engine uses the interface name from the NDE source device using SNMP query. See the "interface-name-key" section on page 4-10 for additional information.

The default configuration of this feature is **/opt/CSCOnfc/config/nfcifname.xml**. You must to configure the SNMP read-only community string of a NDE source device if it differs from the default value **public**. The syntax of this configuration is as follows:

<comm-string device="1.1.1.1">mystring</comm-string>

In order to reduce the amount of SNMP queries and enhance the application performance, CNS NetFlow Collection Engine caches the SNMP query results in memory. You may change the cache refresh period with the following XML:

```
  <!-- Interface Name Cache refresh period (in milliseconds). Default is 30
minutes -->
  <refresh-period>1800000</refresh-period>
```

Cisco CNS NetFlow Collection Engine, 5.0.2 offers the option to retrieve SNMP **ifName** as the mapped interface name, instead of the default **ifDescr**. In order to retrieve **ifName**, change the value in the **<mib-object-to-query>** element to:

<mib-object-to-query>ifName</mib-object-to-query>

Currently, **ifDescr** and **ifName** are the only two valid values for this setting.

After changes are made in the **nfcifname.xml** file, restart CNS NetFlow Collection Engine for these changes to take effect.

Troubleshooting CNS NetFlow Collection Engine

This appendix provides helpful information and procedures in case you encounter problems while using CNS NetFlow Collection Engine.

This appendix includes the following sections:

- "Using the nfcollector status Command" section on page A-1
- "Using the show-tech Command to Capture Troubleshooting Information" section on page A-2
- "CNS NetFlow Collection Engine Tools and Utilities" section on page A-2
- "Solving CNS NetFlow Collection Engine Problems" section on page A-5
- "Web-based UI Troubleshooting Tips" section on page A-9

Using the nfcollector status Command

The **nfcollector status** command provides an easy way to determine which processes are running (or not running). To invoke the **nfcollector status** command, enter the following command line at the UNIX prompt:

$ $NFC_DIR/bin/nfcollector status

When invoked, the **nfcollector status** command displays status information about CNS NetFlow Collection Engine, as in the following example:

```
Running and not running processes:
nfcxml: Running (pid: 1126)
collection: Running (pid: 1128)
re: Not Running
web: Running (pid: 1132)
```

Note If the **nfcollector status** command indicates that a process is stopped, there may be a problem with the CNS NetFlow Collection Engine workstation. See the "Starting the CNS NetFlow Collection Engine User Interface" section on page 2-1 for information on how to start CNS NetFlow Collection Engine processes.

Using the show-tech Command to Capture Troubleshooting Information

The **show-tech** command provides an easy way to generate all the debugging information necessary for support and troubleshooting purposes. To invoke the show-tech command, enter the following command line at the UNIX prompt:

$ $NFC_DIR/bin/nfcollector show-tech

Note To capture running configuration information, you should invoke the **show-tech** command while CNS NetFlow Collection Engine is running.

When invoked, the **show-tech** command creates a log file named **show-tech.log** in the **$NFC_DIR/logs** directory.

CNS NetFlow Collection Engine Tools and Utilities

The utilities described in this section are typically used to troubleshoot CNS NetFlow Collection Engine operation by providing a way to capture and play back received NetFlow data. The process emulates a Cisco export device generating NetFlow data through the NetFlow data export feature. The utilities are available in the **$NFC_DIR/tools** directory and include the following:

- fdcount Utility
- ndeget Utility
- showpacketlog Utility
- get_bgp_rib Utilityfdget Utility
- fdplayback Utility
- nfc_gunzip Utility
- nfc_bin_to_ascii Utility

fdcount Utility

The **fdcount** utility listens to a user-specified UDP port, samples a user-specified number of incoming datagrams, and calculates the average incoming rate. Enter:

```
$NFC_DIR/tools/fdcount [-p UDP-port] [-c count] [-s socket-buffer]
```

where:

-p *UDP-port*	UDP port number on which flows are to be received. The default is 9991.
-c *count*	Number of flows to sample before calculating the incoming rate. The default is 100.
-s *socket-buffer*	Receive socket buffer size, in bytes. The default is 90000 bytes.

ndeget Utility

The **ndeget** utility listens to a user-specified UDP port to receive flow data and prints the contents of the received flow packets to the standard output. This is intended to replace the fdget utility, which is still included for backwards compatibility. Unlike fdget, ndeget can display the contents of NetFlow version 9 packets. Enter:

```
$NFC_DIR/tools/ndeget.sh -port port [-hex] [-maxpacketlen length]
```

where:

-port port	UDP port number on which flows are to be received.
-hex	Optionally display a hex dump of the contents of packets.
-maxpacketlen len	Optionally change the size of the packet buffer.

showpacketlog Utility

The **showpacketlog** utility is similar to the **ndeget** utility, and displays the contents of a packet log file if the packet log option for an aggregator was enabled. Enter:

```
$NFC_DIR/tools/showpacketlog.sh [file ...]
```

where:

file	One or more packet log files; or reads from standard input if no file is specified.

get_bgp_rib Utility

The **get_bgp_rib** utility displays the contents of the CNS NetFlow Collection Engine BGP Peer's routing information base. Enter:

```
$NFC_DIR/tools/get_bgp_rib.sh [-p port ] [ -x ]
```

where:

-p port	Optionally change the port used for contacting the BGP peer.
-x	Optionally display the result as XML.

fdget Utility

The **fdget** utility listens to a user-specified UDP port to receive flow data and prints some of the fields from the received flow packets to the standard output. One use of this capability is to print flow data sent by the **fdplayback** utility. Enter:

```
$NFC_DIR/tools/fdget [-p UDP-port] [-s socket-buffer] [-a]
```

where:

-p *UDP-port*	UDP port number on which flows are to be received. The default is 9991.
-s *socket-buffer*	Receive socket buffer size, in bytes. The default is 90000 bytes. This argument and value determine how many datagrams the kernel stores in this buffer as datagrams come in from the network. The larger the buffer, the more time **fdget** has to consume data from the buffer before the buffer overflows. If the buffer overflows, datagrams are lost.
-a	Print an acknowledgment only. The default is to print the content of flows. Using **-a** means print only an acknowledgment for each datagram received rather than the content of the datagram.

fdplayback Utility

The **fdplayback** utility reads a data file of NetFlow data created by CNS NetFlow Collection Engine or some other tool and sends the flow data to a user-specified destination. Enter:

```
$NFC_DIR/tools/fdplayback [-f datafile] [-d IP-address] [-p UDP-port] [-i delay]
[-b burst] [-s socket-buffer] [-t flows]
```

where:

-f *datafile*	Name of data file to play back to the user-specified destination (defined by IP address and UDP port number).
-d *IP-address*	Destination IP address.
-p *UDP-port*	Destination UDP port number. The default is 9991.
-i *delay*	Delay (in milliseconds) between datagrams. The default is 1000. The longer the delay, the more separation there is between datagrams being sent to the receiving destination.
-b *burst*	Number of flows sent in each burst. The default is 10. This argument is used in conjunction with **-i** to control the speed and "burstiness" of the playback.
-s *socket-buffer*	Receive socket buffer size, in bytes. The default is 90000 bytes.
-t *flows*	Number of flows to play back in this session. The default is all flows in the data file. If the data file contains 1000 datagrams and you set **-t** to 1, **fdplayback** only sends one datagram.

nfc_gunzip Utility

The **nfc_gunzip** utility is used to uncompress CNS NetFlow Collection Engine data files that are created with the compression option set to yes. Compressed files are identified with a **.gz** extension. If the compressed file is in binary format, the extension is **.bin.gz**. See the "Creating an Aggregator" section on page 4-16 for details on these file creation options. To use this utility enter:

```
$NFC_DIR/tools/nfc_gunzip filename
```

nfc_bin_to_ascii Utility

The **nfc_bin_to_ascii** converter utility is used to convert binary format data files to ASCII format data files. Binary data files are identified with a **.bin** extension. If compression is applied to the file, it is identified with a **.bin.gz** extension. See the "Creating an Aggregator" section on page 4-16 for details on these file creation options. To use this utility enter:

```
$NFC_DIR/tools/nfc_bin_to_ascii filename "delimiter"
```

Note The **delimiter** option can be the "," or "|" characters. Quotes are required in the **delimiter** parameter. If no delimiter is used, the "|" character is used by default.

Solving CNS NetFlow Collection Engine Problems

This section discusses some basic problems that you might encounter while attempting to run CNS NetFlow Collection Engine.

Symptom CNS NetFlow Collection Engine data files are not being written to the directory specified in the **output-base-dir** aggregator attribute.

Possible Cause Either the **output-base-dir** aggregator attribute process does not have the appropriate permission settings, or the **max-disk-usage-megabytes** aggregator attribute value has been exceeded.

Recommended Action Look at the **nfc.log** file to find the exact cause. If the problem is permission settings, fix the permission settings and try again. If the problem is related to the **max-disk-usage-megabytes** setting, increase the limit (if acceptable). You might need to make more disk space available in this partition.

Symptom The export device is exporting NetFlow data to a port, but CNS NetFlow Collection Engine does not see any data.

Possible Cause Check the **nfc.log** file for an error message about not being able to bind to that UDP port. If you find such a message, some other application is using that port.

Recommended Action Verify that the export device is not using a reserved port number in its attempt to export data to CNS NetFlow Collection Engine. Use an unreserved port number in the range 1024 to 65535 (for example, 9995 or 9996) to export data to CNS NetFlow Collection Engine.

Symptom During installation on a Solaris system, an error is encountered and CNS NetFlow Collection Engine does not finish installing.

Possible Cause The system is running Solaris 7 or lower.

Recommended Action Use a system running Solaris 8 or Solaris 9. Solaris 7 or lower is not supported.

Symptom During installation on an HP-UX system, an error is encountered and CNS NetFlow Collection Engine does not finish installing.

Possible Cause The system is running HP-UX Version 10.20 or another unsupported HP-UX version.

Recommended Action Use a system running HP-UX version 11.x. All other HP-UX versions are not supported.

Symptom During installation on a Linux system, an error is encountered and CNS NetFlow Collection Engine does not finish installing.

Possible Cause The system is not running Red Hat Enterprise 2.1 or 3 ES Linux.

Recommended Action Use a system running Red Hat Enterprise 2.1 or 3 ES Linux.

Symptom *nfcollector start all* fails to start any processes.

Possible Cause Running as some user other than the owner of CNS NetFlow Collection Engine files and processes, and thus logs can not be written due to permission problems.

Recommended Action Running as the owner of CNS NetFlow Collection Engine.

Symptom Collection process fails to start.

Possible Cause Invalid XML in configuration file **nfc-config.xml**.

Recommended Action Check the **nfc.log** file. Identify the log message corresponding to the invalid XML and fix it. Stop the collector process and use the **fdget** tool to verify that NDE packets are being received.

Symptom There are incoming NDE packets but no aggregation results are output.

Possible Cause Field required by aggregation scheme is missing in NDE flow records.

Recommended Action Check **nfc.log**. If it contains information about missing fields, make sure that the device configuration is correct so that the collector gets the NDE containing those missing fields.

Symptom It takes a long time for results to be generated when user scripts are associated with the aggregator.

Possible Cause User scripts sometimes consume significant amount of time for each output file, based on our experience with customer issues.

Recommended Action Check what the user scripts actually do and cut unnecessary post processing.

Symptom Can not generate a report using a specific aggregator.

Possible Cause Trying to report on an aggregator producing binary output or ASCII output without an XML header.

Recommended Action Run report on an aggregator that produces ASCII output with an XML header.

Symptom The navigation tree is empty in web-based GUI.

Possible Cause 1) Collector is not running; 2) CNS/XML interface is not running; or 3) the subject parameter of the **InitServlet** does not match that of the CNS/XML interface.

Recommended Action Make sure the collector and CNS/XML interface are running. Make sure the subject parameter of the **InitServlet** matches that of the CNS/XML interface.

Symptom Warning log in **NFC_DIR/logs/nfc.log**:

> *aggregator*: field not found in flow: *field-name*, id=*field-id*, NDE version=*nde-version*, template id=*template-id*

Possible Cause The NDE received by *aggregator* does not contain the field *field-name* that is referenced by the selected aggregation scheme.

Recommended Action First determine what fields are available in NDE flows for version *nde-version*. For *nde-version* 1-8, refer to <fixed-flow-packet-types> element for this version in **NFC_DIR/config/nfc-config-predefined.xml**, which lists each field for each version. For *nde-version* 9, refer to the previous log in **NFC_DIR/logs/nfc.log** for this *template-id* that indicates which fields correspond with the template received:

> New data template from *router-address*, id=*template-id*, fields=*field-count*
> field id=*field-id* (*field-name*), offset=*offset-in-flow*, len=*length-in-flow*
> field id=*field-id* (*field-name*), offset=*offset-in-flow*, len=*length-in-flow*
> ...

After determining what fields are available in the NDE received by this aggregator, either specify a new aggregation scheme, or update the specified aggregation scheme to no longer reference the missing field.

If you wish to aggregate packets that are missing one or more fields of interest, starting in CNS NetFlow Collection Engine, 5.0.2 you can also set the key builder attribute **is-null-allowed** to **true**. In this case no warning is written and the column value in CNS NetFlow Collection Engine output files is empty. See Chapter 4, “Customizing the CNS NetFlow Collection Engine” for additional information.

Symptom Warning log in **NFC_DIR/logs/nfc.log**:

> Received packet from an NDE source for which access is denied: nde-source-address.

Possible Cause An access list is configured that does not allow packets from this address.

Recommended Action Configure the device to not send packets to CNS NetFlow Collection Engine, or update the access list entry for this device. See the “Creating Access Lists” section on page 4-24 for more information.

Symptom Error log in **NFC_DIR/logs/nfc.log** related to memory:

Memory was exhausted while processing a flow.
Memory was exhausted while writing output for *aggregator*.

Possible Cause The collector process has insufficient memory for the amount of data that it is receiving.

Recommended Action Memory allocated to various CNS NetFlow Collection Engine subsystems can be adjusted as described in the "Tuning Memory Usage" section on page 4-26.

Symptom An error log in **NFC_DIR/logs/nfc.log** indicating that a file I/O error has occurred:

Error writing output file...
Error updating filesready file...

Possible Cause File errors may result if the file system containing **/opt/CSCOnfc** is full, or in case of a permission problem the wrong user started CNS NetFlow Collection Engine.

Recommended Action Check file system capacity (for example, **df -k /opt/CSCOnfc**). Verify that the user that started CNS NetFlow Collection Engine is the same user that owns directories under **/opt/CSCOnfc**. If a specific file is named in the log, check permissions and ownership of that file.

Symptom Error log in **NFC_DIR/logs/nfc.log** or other collector log file related to memory:

Output for aggregator *ID* has been queued multiple times without being written. Collector memory may be exhausted if this continues in subsequent aggregation periods.

Memory was exhausted while writing output for *file*. Output files for this aggregator in this period may not be complete.

Memory was exhausted while processing a flow.

I/O error when starting *program*: Not enough space, exiting.

java.io.IOException: Not enough space

Possible Cause The collector process has insufficient memory for the amount of data that it is receiving.

Recommended Action Memory-related issues can be addressed in one or more of the following ways:

- If there is sufficient system memory, the memory allocated to various CNS NetFlow Collection Engine subsystems can be adjusted as described in the "Tuning Memory Usage" section on page 4-26.
- Decrease the number of keys and values in aggregation schemes and in reports; decrease the number of active aggregators.
- Decrease the number of devices sending data to the collector.
- Enable NetFlow sampling and/or router-based aggregation on export devices.

Symptom Log into **NFC_DIR/logs/nfcudprdr.log**:

ld.so.1: /opt/CSCOnfc/bin/UDPFlowReaderProgram: fatal: libCstd.so.1: open failed: No such file or directory

Possible Cause A required Solaris patch has not been installed on the system.

Recommended Action Install the required Solaris patch.

Symptom Tools in **/opt/CSCOnfc/tools** fail to start on a Red Hat Enterprise 3 Linux platform.

Possible Cause The required RPM compat-libstdc++ has not been installed on the system.

Recommended Action Install the RPM as described in the section"Installing on a Red Hat Enterprise Linux Platform" in the Cisco CNS NetFlow Collection Engine Installation and Configuration Guide.

Web-based UI Troubleshooting Tips

Here are some troubleshooting tips in case you have problems with the web-based UI.

Table A-1 Troubleshooting Tips

Problem	Suggestion
The navigation tree is empty	Make sure the collector and CNS/XML interface are running. Make sure the subject parameter of the **InitServlet** matches that of the CNS/XML interface.
Can not generate a report using a specific aggregator	Only aggregators that have a single ASCII writer can be used to generate reports.

NetFlow Export Datagram Formats

NetFlow exports flow information in UDP datagrams in one of five formats:

- Version 1
- Version 5
- Version 7
- Version 8
- Version 9

Version 1 (V1) is the original format supported in the initial NetFlow releases.Version 5 (V5) is an enhancement that adds Border Gateway Protocol (BGP) autonomous system information and flow sequence numbers.Version 7 (V7) is an enhancement that exclusively supports NetFlow with Cisco Catalyst 5000 series switches equipped with a NetFlow feature card (NFFC). V7 is not compatible with Cisco routers.Version 8 (V8) is an enhancement that adds router-based aggregation schemes. Version 9 is an enhancement to support different technologies such as Multicast, Internet Protocol Security (IPSec), and Multi Protocol Label Switching (MPLS). Version 9 is not compatible with previous versions of CNS NetFlow Collection Engine.

Versions 2, 3, 4, and 6 either were not released or are not supported by CNS NetFlow Collection Engine.

This appendix describes these formats in the following sections:

- Versions 1, 5, 7 and 8
- Version 9

Versions 1, 5, 7 and 8

In Versions 1, 5, and 7, the datagram consists of a header and one or more flow records. The first field of the header contains the version number of the export datagram. Typically, a receiving application that accepts any of the format versions allocates a buffer large enough for the largest possible datagram from any of the format versions and then uses the header to determine how to interpret the datagram. The second field in the header contains the number of records in the datagram and should be used to search through the records.

All fields described in the format version tables are in network byte order.

- Table B-1 and Table B-2 describe the V1 header and flow record format, respectively
- Table B-3 and Table B-4 describe the V5 header and flow record format, respectively
- Table B-5 and Table B-6 describe the V7 header and flow record format, respectively

- Table B-7 describes the V8 header format
- Table B-8 describes the V8 RouterAS flow record format
- Table B-9 describes the V8 RouterProtoPort flow record
- Table B-10 describes the V8 RouterDstPrefix flow record
- Table B-11 describes the RouterSrcPrefix flow record
- Table B-12 describes the RouterPrefix flow record format
- Table B-13 describes the TosAS flow record format
- Table B-14 describes the TosProtoPort flow record format
- Table B-15 describes the PrePortProtocol flow record format
- Table B-16 describes the TosSrcPrefix flow record format
- Table B-17 describes the TosDstPrefix flow record format
- Table B-18 describes the TosPrefix flow record format
- Table B-19 describes the DestOnly flow record format
- Table B-20 describes the SrcDst flow record format
- Table B-21 describes the FullFlow flow record format.

Note V8 data consists of header information that follows the same format as the other versions. However, the V8 flow record formats are separated based on the aggregation schemes that support router-based aggregation. Instead of one flow record table, you see five tables that describe the V8 flow record format for each individual aggregation scheme.

We recommend that receiving applications perform a *sanity check* on datagrams to ensure that the datagrams are from a valid NetFlow source. You should first check the size of the datagram to verify that it is at least long enough to contain the version and count fields. You should next verify that the version is valid (1, 5, 7, or 8) and that the number of received bytes is enough for the header and count flow records (using the appropriate version).

Because NetFlow export uses UDP to send export datagrams, it is possible for datagrams to be lost. To determine whether flow export information has been lost, Version 5, Version 7, and Version 8 headers contain a flow sequence number. The sequence number is equal to the sequence number of the previous datagram plus the number of flows in the previous datagram. After receiving a new datagram, the receiving application can subtract the expected sequence number from the sequence number in the header to derive the number of missed flows.

Datagram format Version 8 offers five router-based aggregation schemes allowing you to summarize CNS NetFlow Collection Engine export data on the router before the data is exported to the CNS NetFlow Collection Engine. The result is lower bandwidth requirements and reduced platform requirements for NetFlow data collection devices.

Router-based aggregation enables on-router aggregation by maintaining one or more extra NetFlow caches with different combinations of fields that determine which traditional flows are grouped together. These extra caches are called aggregation caches. As flows expire from the main flow cache, they are added to each enabled aggregation cache. The normal flow ager process runs on each active aggregation cache the same way it runs on the main cache. On-demand aging is also supported.

Table B-1 describes the V1 header format.

Table B-1 Version 1 Header Format

Bytes	Contents	Description
0-1	version	NetFlow export format version number
2-3	count	Number of flows exported in this packet (1-24)
4-7	SysUptime	Current time in milliseconds since the export device booted
8-11	unix_secs	Current count of seconds since 0000 UTC 1970
12-15	unix_nsecs	Residual nanoseconds since 0000 UTC 1970

Table B-2 describes the V1 flow record format.

Table B-2 Version 1 Flow Record Format

Bytes	Contents	Description
0-3	srcaddr	Source IP address
4-7	dstaddr	Destination IP address
8-11	nexthop	IP address of next hop router
12-13	input	SNMP index of input interface
14-15	output	SNMP index of output interface
16-19	dPkts	Packets in the flow
20-23	dOctets	Total number of Layer 3 bytes in the packets of the flow
24-27	First	SysUptime at start of flow
28-31	Last	SysUptime at the time the last packet of the flow was received
32-33	srcport	TCP/UDP source port number or equivalent
34-35	dstport	TCP/UDP destination port number or equivalent
36-37	pad1	Unused (zero) bytes
38	prot	IP protocol type (for example, TCP = 6; UDP = 17)
39	tos	IP type of service (ToS)
40	flags	Cumulative OR of TCP flags
41-43	pad1, pad2, pad3	Unused (zero) bytes
44-47	reserved	Unused (zero) bytes

Table B-3 describes the V5 header format.

Table B-3 Version 5 Header Format

Bytes	Contents	Description
0-1	version	NetFlow export format version number
2-3	count	Number of flows exported in this packet (1-30)
4-7	SysUptime	Current time in milliseconds since the export device booted

Table B-3 Version 5 Header Format (continued)

Bytes	Contents	Description
8-11	unix_secs	Current count of seconds since 0000 UTC 1970
12-15	unix_nsecs	Residual nanoseconds since 0000 UTC 1970
16-19	flow_sequence	Sequence counter of total flows seen
20	engine_type	Type of flow-switching engine
21	engine_id	Slot number of the flow-switching engine
22-23	sampling_interval	First two bits hold the sampling mode; remaining 14 bits hold value of sampling interval

Table B-4 describe the V5 flow record format.

Table B-4 Version 5 Flow Record Format

Bytes	Contents	Description
0-3	srcaddr	Source IP address
4-7	dstaddr	Destination IP address
8-11	nexthop	IP address of next hop router
12-13	input	SNMP index of input interface
14-15	output	SNMP index of output interface
16-19	dPkts	Packets in the flow
20-23	dOctets	Total number of Layer 3 bytes in the packets of the flow
24-27	First	SysUptime at start of flow
28-31	Last	SysUptime at the time the last packet of the flow was received
32-33	srcport	TCP/UDP source port number or equivalent
34-35	dstport	TCP/UDP destination port number or equivalent
36	pad1	Unused (zero) bytes
37	tcp_flags	Cumulative OR of TCP flags
38	prot	IP protocol type (for example, TCP = 6; UDP = 17)
39	tos	IP type of service (ToS)
40-41	src_as	Autonomous system number of the source, either origin or peer
42-43	dst_as	Autonomous system number of the destination, either origin or peer
44	src_mask	Source address prefix mask bits
45	dst_mask	Destination address prefix mask bits
46-47	pad2	Unused (zero) bytes

Table B-5 describes the V7 header format.

Table B-5 Version 7 (Catalyst 5000) Header Format

Bytes	Contents	Description
0-1	version	NetFlow export format version number
2-3	count	Number of flows exported in this flow frame (protocol data unit, or PDU)
4-7	SysUptime	Current time in milliseconds since the export device booted
8-11	unix_secs	Current seconds since 0000 UTC 1970
12-15	unix_nsecs	Residual nanoseconds since 0000 UTC 1970
16-19	flow_sequence	Sequence counter of total flows seen
20-23	reserved	Unused (zero) bytes

Table B-6 describe the V7 flow record format.

Table B-6 Version 7 (Catalyst 5000) Flow Record Format

Bytes	Contents	Description
0-3	srcaddr	Source IP address; in case of destination-only flows, set to zero.
4-7	dstaddr	Destination IP address.
8-11	nexthop	Next hop router; always set to zero.
12-13	input	SNMP index of input interface; always set to zero.
14-15	output	SNMP index of output interface.
16-19	dPkts	Packets in the flow.
20-23	dOctets	Total number of Layer 3 bytes in the packets of the flow.
24-27	First	SysUptime, in milliseconds, at start of flow.
28-31	Last	SysUptime, in milliseconds, at the time the last packet of the flow was received.
32-33	srcport	TCP/UDP source port number; set to zero if flow mask is destination-only or source-destination.
34-35	dstport	TCP/UDP destination port number; set to zero if flow mask is destination-only or source-destination.
36	flags	Flags indicating, among other things, what flow fields are invalid.
37	tcp_flags	TCP flags; always set to zero.
38	prot	IP protocol type (for example, TCP = 6; UDP = 17); set to zero if flow mask is destination-only or source-destination.
39	tos	IP type of service; switch sets it to the ToS of the first packet of the flow.
40-41	src_as	Source autonomous system number, either origin or peer; always set to zero.
42-43	dst_as	Destination autonomous system number, either origin or peer; always set to zero.

Table B-6 Version 7 (Catalyst 5000) Flow Record Format (continued)

Bytes	Contents	Description
44	src_mask	Source address prefix mask; always set to zero.
45	dst_mask	Destination address prefix mask; always set to zero.
46-47	flags	Flags indicating, among other things, what flows are invalid.
48-51	router_sc	IP address of the router that is bypassed by the Catalyst 5000 series switch. This is the same address the router uses when it sends NetFlow export packets. This IP address is propagated to all switches bypassing the router through the FCP protocol.

Table B-7 describes the V8 header format.

Note Version 7 AS information is not supported in current implementations of the Catalyst 5000 series switch.

Table B-7 Version 8 Header Format

Bytes	Contents	Description
0-1	version	NetFlow export format version number
2-3	count	Number of flows exported in this flow frame (protocol data unit, or PDU)
4-7	SysUptime	Current time in milliseconds since the export device booted
8-11	unix_secs	Current seconds since 0000 UTC 1970
12-15	unix_nsecs	Residual nanoseconds since 0000 UTC 1970
16-19	flow_sequence	Sequence counter of total flows seen
20	engine_type	Type of flow switching engine
21	engine_id	ID number of the flow switching engine
22	aggregation	Aggregation method being used
23	agg_version	Version of the aggregation export
24-27	reserved	Unused (zero) bytes

Table B-8 describes the V8 RouterAS flow record format.

Table B-8 Version 8 RouterAS Flow Record Format

Bytes	Contents	Description
0-3	flows	Number of flows
4-7	dPkts	Packets in the flow
8-11	dOctets	Total number of Layer 3 bytes in the packets of the flow
12-15	First	SysUptime, in seconds, at start of flow
16-19	Last	SysUptime, in seconds, at the time the last packet of the flow was received

Table B-8 Version 8 RouterAS Flow Record Format (continued)

Bytes	Contents	Description
20-21	src_as	Source autonomous system number, either origin or peer; always set to zero
22-23	dst_as	Destination autonomous system number, either origin or peer; always set to zero
24-25	input	SNMP index of input interface; always set to zero
26-27	output	SNMP index of output interface

Table B-9 describes the V8 **RouterProtoPort** flow record.

Table B-9 Version 8 RouterProtoPort Flow Record Format

Bytes	Contents	Description
0-3	flows	Number of flows
4-7	dPkts	Packets in the flow
8-11	dOctets	Total number of Layer 3 bytes in the packets of the flow
12-15	First	SysUptime, in seconds, at start of flow
16-19	Last	SysUptime, in seconds, at the time the last packet of the flow was received
20	prot	IP protocol type (for example, TCP = 6; UDP = 17); set to zero if flow mask is destination-only or source-destination
21	pad	Unused (zero) bytes
22-23	reserved	Unused (zero) bytes
24-25	srcport	TCP/UDP source port number; set to zero if flow mask is destination-only or source-destination
26-27	dstport	TCP/UDP destination port number; set to zero if flow mask is destination-only or source-destination

Table B-10 describes the V8 **RouterDstPrefix** flow record.

Table B-10 Version 8 RouterDstPrefix Flow Record Format

Bytes	Contents	Description
0-3	flows	Number of flows
4-7	dPkts	Packets in the flow
8-11	dOctets	Total number of Layer 3 bytes in the packets of the flow
12-15	First	SysUptime, in seconds, at start of flow
16-19	Last	SysUptime, in seconds, at the time the last packet of the flow was received
20-23	dst_prefix	Destination IP address prefix
24	dst_mask	Destination address prefix mask; always set to zero
25	pad	Unused (zero) bytes

Table B-10 Version 8 RouterDstPrefix Flow Record Format (continued)

Bytes	Contents	Description
26-27	dst_as	Destination autonomous system number, either origin or peer; always set to zero
28-29	output	SNMP index of output interface
30-31	reserved	Unused (zero) bytes

Table B-11 describes the **RouterSrcPrefix** flow record.

Table B-11 Version 8 RouterSrcPrefix Flow Record Format

Bytes	Contents	Description
0-3	flows	Number of flows
4-7	dPkts	Packets in the flow
8-11	dOctets	Total number of Layer 3 bytes in the packets of the flow
12-15	First	SysUptime, in seconds, at start of flow
16-19	Last	SysUptime, in seconds, at the time the last packet of the flow was received
20-23	src_prefix	Source IP address prefix
24	src_mask	Source address prefix mask; always set to zero
25	pad	Unused (zero) bytes
26-27	src_as	Source autonomous system number, either origin or peer; always set to zero
28-29	input	SNMP index of input interface; always set to zero
30-31	reserved	Unused (zero) bytes

Table B-12 describes the **RouterPrefix** flow record format.

Table B-12 Version 8 RouterPrefix Flow Record Format

Bytes	Contents	Description
0-3	flows	Number of flows
4-7	dPkts	Packets in the flow
8-11	dOctets	Total number of Layer 3 bytes in the packets of the flow
12-15	First	SysUptime, in seconds, at start of flow
16-19	Last	SysUptime, in seconds, at the time the last packet of the flow was received
20-23	src_prefix	Source IP address prefix
24-27	dst_prefix	Destination IP address prefix
28	dst_mask	Source address prefix mask; always set to zero
29	src_mask	Destination address prefix mask; always set to zero
30-31	reserved	Unused (zero) bytes

Table B-12 Version 8 RouterPrefix Flow Record Format (continued)

Bytes	Contents	Description
32-33	src_as	Source autonomous system number, either origin or peer; always set to zero
34-35	dst_as	Destination autonomous system number, either origin or peer; always set to zero
36-37	input	SNMP index of input interface; always set to zero
38-39	output	SNMP index of output interface

Table B-13 describes the **TosAS** flow record format.

Table B-13 Version 8 TosAS Record Format

Bytes	Contents	Description
0-3	flows	Number of flows
4-7	dPkts	Packets in the flow
8-11	dOctets	Total number of Layer 3 bytes in the packets of the flow
12-15	First	SysUptime, in seconds, at start of flow
16-19	Last	SysUptime, in seconds, at the time the last packet of the flow was received
20-21	src_as	Source autonomous system number, either origin or peer; always set to zero
22-23	dst_as	Destination autonomous system number, either origin or peer; always set to zero
24-25	input	SNMP index of input interface; always set to zero
26-27	output	SNMP index of output interface
28	tos	Type of service
29	pad	Unused (zero) bytes
30-31	reserved	Unused (zero) bytes

Table B-14 describes the **TosProtoPort** flow record format.

Table B-14 Version 8 TosProtoPort Record Format

Bytes	Contents	Description
0-3	flows	Number of flows
4-7	dPkts	Packets in the flow
8-11	dOctets	Total number of Layer 3 bytes in the packets of the flow
12-15	First	SysUptime, in seconds, at start of flow
16-19	Last	SysUptime, in seconds, at the time the last packet of the flow was received
20	prot	IP protocol type (for example, TCP = 6; UDP = 17); set to zero if flow mask is destination-only or source-destination

Table B-14 Version 8 TosProtoPort Record Format (continued)

Bytes	Contents	Description
21	Tos	IP Type of Service
22-23	reserved	Unused (zero) bytes
24-25	srcport	TCP/UDP source port number; set to zero if flow mask is destination-only or source-destination
26-27	dstport	TCP/UDP destination port number; set to zero if flow mask is destination-only or source-destination
28-29	input	SNMP index of input interface
30-31	output	SNMP index of output interface

Table B-15 describes the **PrePortProtocol** flow record format.

Table B-15 Version 8 PrePortProtocol Record Format

Bytes	Contents	Description
0-3	flows	Number of flows
4-7	dpkts	Packets in the flow
8-11	dOctets	Total number of Layer 3 bytes in the packets of the flow
12-15	First	SysUptime, in seconds, at start of flow
16-19	Last	SysUptime, in seconds, at the time the last packet of the flow was received
20-23	src_prefix	Source IP address prefix
24-27	dst_prefix	Destination IP address prefix
28	dst_mask	Destination address prefix mask
29	src_mask	Source address prefix mask
30	Tos	IP Type of Service
31	prot	IP protocol type (for example, TCP = 6; UDP = 17); set to zero if flow mask is destination-only or source-destination
32-33	srcport	TCP/UDP source port number; set to zero if flow mask is destination-only or source-destination
34-35	dstport	TCP/UDP destination port number; set to zero if flow mask is destination-only or source-destination
36-37	input	SNMP index of input interface
38-39	output	SNMP index of output interface

Table B-16 describes the **TosSrcPrefix** flow record format.

Table B-16 Version 8 TosSrcPrefix Record Format

Bytes	Contents	Description
0-3	flows	Number of flows
4-7	dPkts	Packets in the flow
8-11	dOctets	Total number of Layer 3 bytes in the packets of the flow
12-15	First	SysUptime, in seconds, at start of flow
16-19	Last	SysUptime, in seconds, at the time the last packet of the flow was received
20-23	src_prefix	Source IP address prefix
24	src_mask	Source address prefix mask
25	Tos	IP Type of Service
26-27	src_as	Source autonomous system number, either origin or peer
28-29	input	SNMP index of input interface
30-31	reserved	Reserved for future use

Table B-17 describes the **TosDstPrefix** flow record format.

Table B-17 Version 8 TosDstPrefix Record Format

Bytes	Contents	Description
0-3	flows	Number of flows
4-7	dPkts	Packets in the flow
8-11	dOctets	Total number of Layer 3 bytes in the packets of the flow
12-15	First	SysUptime, in seconds, at start of flow
16-19	Last	SysUptime, in seconds, at the time the last packet of the flow was received
20-23	dst_prefix	Destination IP address prefix
24	dst_mask	Destination address prefix mask
25	Tos	IP Type of Service
26-27	dst_as	Destination autonomous system number, either origin or peer
28-29	output	SNMP index of output interface
30-31	reserved	Unused (zero) bytes

Table B-18 describes the **TosPrefix** flow record format.

Table B-18 Version 8 TosPrefix Record Format

Bytes	Contents	Description
0-3	flows	Number of flows
4-7	dPkts	Packets in the flow

Table B-18 Version 8 TosPrefix Record Format (continued)

Bytes	Contents	Description
8-11	dOctets	Total number of Layer 3 bytes in the packets of the flow
12-15	First	SysUptime, in seconds, at start of flow
16-19	Last	SysUptime, in seconds, at the time the last packet of the flow was received
20-23	src_prefix	Source IP address prefix
24-27	dst_prefix	Destination IP address prefix
28	dst_mask	Destination address prefix mask
29	src_mask	Source address prefix mask
30	Tos	IP Type of Service
31	pad	Unused (zero) bytes
32-33	src_as	Source autonomous system number, either origin or peer
34-35	dst_as	Destination autonomous system number, either origin or peer
36-37	input	SNMP index of input interface
38-39	output	SNMP index of output interface

Table B-19 describes the **DestOnly** flow record format.

Note This flow statistic record is only used in Catalyst 6000 Series **DestOnly** aggregation.

Table B-19 Version 8 DestOnly Record Format

Bytes	Contents	Description
0-3	dstaddr	Destination IP address
4-7	dPkts	Packets in the flow
8-11	dOctets	Total number of Layer 3 bytes in the packets of the flow
12-15	First	SysUptime, in seconds, at start of flow
16-19	Last	SysUptime, in seconds, at the time the last packet of the flow was received
20-21	Output	SNMP index of output interface
22	Tos	IP Type of Service
23	marked_tos	Type of Service of the packets that exceeded the contract
24-27	extraPkts	Packets that exceed the contract
28-31	router_sc	IP address of the router that is bypassed by the Catalyst 5000 series switch. This is the same address the router uses when it sends NetFlow export packets. This IP address is propagated to all switches bypassing the router through the FCP protocol.

Table B-20 describes the **SrcDst** flow record format.

Note This flow statistic record is used in Catalyst 6000 Series only **SrcDst** aggregation.

Table B-20 Version 8 SrcDst Record Format

Bytes	Contents	Description
0-3	dstaddr	Destination IP address
4-7	srcaddr	Source IP address; in case of destination-only flows, set to zero
8-11	dPkts	Packets in the flow
12-15	dOctets	Total number of Layer 3 bytes in the packets of the flow
16-19	First	SysUptime, in seconds, at start of flow
20-23	Last	SysUptime, in seconds, at the time the last packet of the flow was received
24-25	Output	SNMP index of output interface
26-27	Input	SNMP index of input interface
28	Tos	IP Type of Service
29	marked_tos	Type of Service of the packets that exceeded the contract
30-31	reserved	Unused (zero) bytes
32-35	extraPkts	Packets that exceed the contract
36-39	router_sc	IP address of the router that is bypassed by the Catalyst 5000 series switch. This is the same address the router uses when it sends NetFlow export packets. This IP address is propagated to all switches bypassing the router through the FCP protocol.

Table B-21 describes the **FullFlow** flow record format.

Note This flow statistic record is used in Catalyst 6000 Series only **FullFlow** aggregation.

Table B-21 Version 8 FullFlow Record Format

Bytes	Contents	Description
0-3	dstaddr	Destination IP address
4-7	srcaddr	Source IP address; in case of destination-only flows, set to zero
8-9	dstport	TCP/UDP destination port number; set to zero if flow mask is destination-only or source-destination
10-11	srcport	TCP/UDP source port number; set to zero if flow mask is destination-only or source-destination
12-15	dPkts	Packets in the flow
16-19	dOctets	Total number of Layer 3 bytes in the packets of the flow
20-23	First	SysUptime, in seconds, at start of flow

Table B-21 Version 8 FullFlow Record Format (continued)

Bytes	Contents	Description
24-27	Last	SysUptime, in seconds, at the time the last packet of the flow was received
28-29	Output	SNMP index of output interface
30-31	Input	SNMP index of input interface
32	Tos	IP Type of Service
33	prot	IP protocol type (for example, TCP = 6; UDP = 17); set to zero if flow mask is destination-only or source-destination
34	marked_tos	Type of Service of the packets that exceeded the contract
35	pad	Unused (zero) bytes
36-39	extraPkts	Packets that exceed the contract
40-43	router_sc	IP address of the router that is bypassed by the Catalyst 5000 series switch. This is the same address the router uses when it sends NetFlow export packets. This IP address is propagated to all switches bypassing the router through the FCP protocol.

Version 9

The distinguishing feature of the NetFlow Version 9 format is that it is template based. Templates provide an extensible design to the record format, a feature that should allow future enhancements to NetFlow services without requiring concurrent changes to the basic flow-record format.

This section includes the following:

- Table B-22 describes Version 9 export packet
- Table B-23 describes Version 9 header format
- Table B-24 describes Version 9 template FlowSet format
- Table B-25 describes Version 9 field types
- Table B-26 describes Version 9 data FlowSet format
- Table B-27 describes Version 9 option template format

Packet Layout

The NetFlow Version 9 record format consists of a packet header followed by at least one or more template or data FlowSets. A template FlowSet provides a description of the fields that will be present in future data FlowSets. These data FlowSets may occur later within the same export packet or in subsequent export packets.

Template and data FlowSets can be intermingled within a single export packet, as illustrated in Table B-22.

Table B-22 Version 9 Export Packet

Packet Header	Template FlowSet	Data FlowSet	Data FlowSet	.....	Template FlowSet	Data FlowSet

Packet Header Format

The format of the NetFlow Version 9 packet header remains relatively unchanged from that of previous versions. Table B-23 describes the Version 9 header format.

Table B-23 Version 9 Header Format

Bytes	Field Name	Description
0-1	version	NetFlow export format version number; for Version 9 this value is 0x0009.
2-3	count	Number of flow sets exported in this packet, both template and data (1-30).
4-7	SysUptime	Current time in milliseconds since the export device booted.
8-11	unix_secs	Current count of seconds since 0000 UTC 1970.
12-15	package_sequence	Sequence counter of all export packets sent by the export device. Note: This is a change from the Version 5 and Version 8 headers, where this number represented “total flows.”
16-19	source_id	A 32-bit value that is used to guarantee uniqueness for all flows exported from a particular device.

Template FlowSet Format

One of the key elements in the new Version 9 format is the template FlowSet. Templates greatly enhance the flexibility of the NetFlow record format, because they allow a NetFlow collector or display application to process NetFlow data without necessarily knowing the format of the data in advance. Templates are used to describe the type and length of individual fields within subsequent NetFlow data records that match a template ID.

Example B-1 Template FlowSet Format

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
FlowSet ID = 0															
Length															
Template ID															
Field Count															
Field 1 Type															
Field 1 Length															
Field 2 Type															
Field 2 Length															
. . .															
Field *N* Type															
Field *N* Length															
Template ID															
Field Count															
Field 1 Type															
Field 1 Length															
Field 2 Type															
Field 2 Length															
Field 1 Type															
. . .															
Field *N* Type															
Field *N* Length															

84829

Table B-24 describes the Version 9 Template FlowSet format.

Table B-24 Version 9 Template FlowSet Format

Field Name	Description
FlowSet ID	Distinguishes template records from data records. A template record always has a FlowSet ID in the range of 0-255.
Length	Refers to the total length of this FlowSet. Because an individual template FlowSet may contain multiple template IDs, the length value should be used to determine the position of the next FlowSet record, which could be either a template or a data FlowSet. Length is expressed in type/length/value (TLV) format, meaning that the value includes the bytes used for the FlowSet ID and the length bytes themselves, as well as the combined lengths of all template records included in this FlowSet.
Template ID	As a router generates different template FlowSets to match the type of NetFlow data it will be exporting, each template is given a unique ID. This uniqueness is local to the router that generated the template ID. Templates that define data record formats begin numbering at 256, because 0-255 are reserved for FlowSet IDs.
Field Count	The number of fields in this template record. Because a template FlowSet may contain multiple template records, this field allows the parser to determine the end of the current template record and the start of the next.
Field Type	Numeric value that represents the type of the field. The possible values of the field type are vendor specific. Cisco supplied values are consistent across all platforms that support NetFlow Version 9. The currently defined field types are detailed in Table B-25.
Field Length	The length of the Field Type field, in bytes.

Note the following:

- Template IDs are consistent across a router reboot. Template IDs should change only if the configuration of NetFlow on the export device changes.
- Templates periodically expire if they are not refreshed. Templates can be refreshed in two ways. A template can be resent every N number of export packets. A template can also be sent on a timer, so that it is refreshed every N number of minutes. Both options are user configurable.

Table B-25 describes the Version 9 field types.

Table B-25 Version 9 Field Type Definitions

Field Type	Value	Length in bytes	Description
IN_BYTES	1	N	Incoming counter with length N x 8 bits for number of bytes associated with an IP Flow. Default: 4.
IN_PKTS	2	N	Incoming counter with length N x 8 bits for the number of packets associated with an IP Flow. Default: 4.
FLOWS	3	N	Number of flows that were aggregated. Default: 4.
PROTOCOL	4	1	IP protocol byte.
SRC_TOS	5	1	Type of Service byte setting when entering incoming interface.
TCP_FLAGS	6	1	Cumulative of all TCP flags seen for this flow.
L4_SRC_PORT	7	2	TCP/UDP source port number. For example, FTP, Telnet, or equivalent.
IPV4_SRC_ADDR	8	4	IPv4 source address.
SRC_MASK	9	1	The number of contiguous bits in the source address subnet mask. For example, the submask in slash notation.
INPUT_SNMP	10	N	Input interface index. Default: 2 but higher values can be used.
L4_DST_PORT	11	2	TCP/UDP destination port number. For example, FTP, Telnet, or equivalent.
IPV4_DST_HOP	12	4	IPv4 destination address.
DST_MASK	13	1	The number of contiguous bits in the destination address subnet mask, that is the submask in slash notation.
OUTPUT_SNMP	14	N	Output interface index. Default: 2 but higher values can be used.
IPV4_NEXT_HOP	15	4	IPv4 address of next-hop router.
SRC_AS	16	N	Source BGP autonomous system number where N could be 2 or 4. Default: 2.
DST_AS	17	N	Destination BGP autonomous system number where N could be 2 or 4. Default: 2.
BGP_IPV4_NEXT_HOP	18	4	Next-hop router's IP in the BGP domain.
MUL_DST_PKTS	19	N	IP multicast outgoing packet counter with length N x 8 bits for packets associated with the IP Flow. Default: 4.
MUL_DST_BYTES	20	N	IP multicast outgoing byte counter with length N x 8 bits for bytes associated with the IP Flow. Default: 4.

Table B-25 Version 9 Field Type Definitions (continued)

Field Type	Value	Length in bytes	Description
LAST_SWITCHED	21	4	System uptime at which the last packet of this flow was switched.
FIRST_SWITCHED	22	4	System uptime at which the first packet of this flow was switched.
OUT_BYTES	23	N	Outgoing counter with length N x 8 bits for the number of bytes associated with an IP Flow. Default: 4.
OUT_PKTS	24	N	Outgoing counter with length N x 8 bits for the number of packets associated with an IP Flow. Default: 4.
MIN_PKT_LNGTH	25	2	Minimum IP packet length on incoming packets of the flow.
MAX_PKT_LNGTH	26	2	Maximum IP packet length on incoming packets of the flow.
IPV6_SRC_ADDR	27	16	IPv6 Source Address.
IPV6_DST_ADDR	28	16	IPv6 Destination Address.
IPV6_SRC_MASK	29	1	Length of the IPv6 source mask in contiguous bits.
IPV6_DST_MASK	30	1	Length of the IPv6 destination mask in contiguous bits.
IPV6_FLOW_LABEL	31	3	IPv6 flow label as per RFC 2460 definition.
ICMP_TYPE	32	2	Internet Control Message Protocol (ICMP) packet type; reported as ((ICMP Type * 256) + ICMP code.)
MUL_IGMP_TYPE	33	1	Internet Group Management Protocol (IGMP) packet type.
SAMPLING_INTERVAL	34	4	When using sampled NetFlow, the rate at which packets are sampled. For example, a value of 100 indicates that one of every 100 packets is sampled
SAMPLING_ALGORITHM	35	1	The type of algorithm used for sampled NetFlow: 0x01 Deterministic Sampling, 0x02 Random Sampling.
FLOW_ACTIVE_TIMEOUT	36	2	Timeout value (in seconds) for active flow entries in the NetFlow cache.
FLOW_INACTIVE_TIMEOUT	37	2	Timeout value (in seconds) for inactive flow entries in the NetFlow cache.
ENGINE_TYPE	38	1	Type of flow switching engine: RP = 0, VIP/Linecard = 1.
ENGINE_ID	39	1	ID number of the flow switching engine.

Table B-25 Version 9 Field Type Definitions (continued)

Field Type	Value	Length in bytes	Description
TOTAL_BYTES_EXP	40	N	Counter with length N x 8 bits for bytes for the number of bytes exported by the Observation Domain. Default: 4.
TOTAL_PKTS_EXP	41	N	Counter with length N x 8 bits for bytes for the number of packets exported by the Observation Domain. Default: 4.
TOTAL_FLOWS_EXP	42	N	Counter with length N x 8 bits for bytes for the number of flows exported by the Observation Domain. Default: 4.
* Vendor Proprietary*	43		
IPV4_SRC_PREFIX	44	4	IPv4 source address prefix (specific for Catalyst architecture).
IPV4_DST_PREFIX	45	4	IPv4 destination address prefix (specific for Catalyst architecture).
MPLS_TOP_LABEL_TYPE	46	1	MPLS Top Label Type: 0x00 UNKNOWN 0x01 TE-MIDPT 0x02 ATOM 0x03 VPN 0x04 BGP 0x05 LDP.
MPLS_TOP_LABEL_IP_ADDR	47	4	Forwarding Equivalent Class corresponding to the MPLS Top Label.
FLOW_SAMPLER_ID	48	1	Identifier shown in show flow-sampler.
FLOW_SAMPLER_MODE	49	1	The type of algorithm used for sampling data: 0x02 random sampling. Use in connection with **FLOW_SAMPLER_MODE**.
FLOW_SAMPLER_RANDOM_ INTERVAL	50	4	Packet interval at which to sample. Use in connection with **FLOW_SAMPLER_MODE**.
* Vendor Proprietary*	51		
MIN_TTL	52	1	Minimum TTL on incoming packets of the flow.
MAX_TTL	53	1	Maximum TTL on incoming packets of the flow.
IPV4_IDENT	54	2	The IP v4 identification field.
DST_TOS	55	1	Type of Service byte setting when exiting outgoing interface.
IN_SRC_MAC	56	6	Incoming source MAC address.
OUT_DST_MAC	57	6	Outgoing destination MAC address.
SRC_VLAN	58	2	Virtual LAN identifier associated with ingress interface.
DST_VLAN	59	2	Virtual LAN identifier associated with egress interface.

Table B-25 Version 9 Field Type Definitions (continued)

Field Type	Value	Length in bytes	Description
IP_PROTOCOL_VERSION	60	1	Internet Protocol Version Set to 4 for IPv4, set to 6 for IPv6. If not present in the template, then version 4 is assumed.
DIRECTION	61	1	Flow direction: 0 - ingress flow, 1 - egress flow.
IPV6_NEXT_HOP	62	16	IPv6 address of the next-hop router.
BPG_IPV6_NEXT_HOP	63	16	Next-hop router in the BGP domain.
IPV6_OPTION_HEADERS	64	4	Bit-encoded field identifying IPv6 option headers found in the flow.
* Vendor Proprietary*	65		
* Vendor Proprietary*	66		
* Vendor Proprietary*	67		
* Vendor Proprietary*	68		
* Vendor Proprietary*	69		
MPLS_LABEL_1	70	3	MPLS label at position 1 in the stack. This comprises 20 bits of MPLS label, 3 EXP (experimental) bits and 1 S (end-of-stack) bit.
MPLS_LABEL_2	71	3	MPLS label at position 2 in the stack. This comprises 20 bits of MPLS label, 3 EXP (experimental) bits and 1 S (end-of-stack) bit.
MPLS_LABEL_3	72	3	MPLS label at position 3 in the stack. This comprises 20 bits of MPLS label, 3 EXP (experimental) bits and 1 S (end-of-stack) bit.
MPLS_LABEL_4	73	3	MPLS label at position 4 in the stack. This comprises 20 bits of MPLS label, 3 EXP (experimental) bits and 1 S (end-of-stack) bit.
MPLS_LABEL_5	74	3	MPLS label at position 5 in the stack. This comprises 20 bits of MPLS label, 3 EXP (experimental) bits and 1 S (end-of-stack) bit.
MPLS_LABEL_6	75	3	MPLS label at position 6 in the stack. This comprises 20 bits of MPLS label, 3 EXP (experimental) bits and 1 S (end-of-stack) bit.
MPLS_LABEL_7	76	3	MPLS label at position 7 in the stack. This comprises 20 bits of MPLS label, 3 EXP (experimental) bits and 1 S (end-of-stack) bit.

Table B-25 Version 9 Field Type Definitions (continued)

Field Type	Value	Length in bytes	Description
MPLS_LABEL_8	77	3	MPLS label at position 8 in the stack. This comprises 20 bits of MPLS label, 3 EXP (experimental) bits and 1 S (end-of-stack) bit.
MPLS_LABEL_9	78	3	MPLS label at position 9 in the stack. This comprises 20 bits of MPLS label, 3 EXP (experimental) bits and 1 S (end-of-stack) bit.
MPLS_LABEL_10	79	3	MPLS label at position 10 in the stack. This comprises 20 bits of MPLS label, 3 EXP (experimental) bits and 1 S (end-of-stack) bit.
IN_DST_MAC	80	6	Incoming destination MAC address
OUT_SRC_MAC	81	6	Outgoing source MAC address
IF_NAME	82	N	Shortened interface name. For example, FE1/0. Default specified in template.
IF_DESC	83	N	Full interface name. For example, FastEthernet 1/0. Default specified in template.
SAMPLER_NAME	84	N	Name of the flow sampler. Default specified in template.
IN_ PERMANENT _BYTES	85	N	Running byte counter for a permanent flow. Default specified in template.
IN_ PERMANENT _PKTS	86	N	Running packet counter for a permanent flow. Default: 4.
* Vendor Proprietary*	87		

Table B-25 Version 9 Field Type Definitions (continued)

Field Type	Value	Length in bytes	Description
FRAGMENT_OFFSET	88	2	The fragment-offset value from fragmented IP packets
FORWARDING STATUS	89	1	Forwarding status with values: – Unknown 0 – Normal forwarding 1, – Forward fragmented 2 – Drop 16 – Drop ACL Deny 17 – Drop ACL drop 18 – Drop Unroutable 19 – Drop Adjacency 20 – Drop Fragmentation & DF set 21 – Drop Bad header checksum 22 – Drop Bad total Length 23 – Drop Bad Header Length 24 – Drop bad TTL 25 – Drop Policer 26 – Drop WRED 27 – Drop RPF 28 – Drop For us 29 – Drop Bad output interface 30 – Drop Hardware 31 – Terminate 128 – Terminate Punt Adjacency 129 – Terminate Incomplete Adjacency 130 – Terminate For us 131

Data FlowSet Format

The following is an example of the Data FlowSet format.

Example B-2 Data FlowSet Format

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
FlowSet ID = Template ID															
Length															
Record 1 - Field 1 value															
Record 1 - Field 2 value															
Record 1 - Field 3 value															
Record 1 - Field 4 value															
. . .															
Record 1 - Field N value															
Record 2 - Field 1 value															
Record 2 - Field 2 value															
Record 2 - Field 3 value															
. . .															
Record 2 - Field N value															
. . .															
Padding															

84831

Table B-26 describes the Version 9 Data FlowSet format.

Table B-26 Version 9 Data FlowSet Format

Field Name	Description
FlowSet ID = Template ID	A FlowSet ID precedes each group of records within a Version 9 data FlowSet. The FlowSet ID maps to a (previously received) template ID. The collector and display applications should use the FlowSet ID to map the appropriate type and length to any field values that follow.
Length	The length of the data FlowSet. Length is expressed in TLV format, meaning that the value includes the bytes used for the FlowSet ID and the length bytes themselves, as well as the combined lengths of any included data records.
Record N - Field N	The remainder of the Version 9 data FlowSet is a collection of field values. The type and length of the fields have been previously defined in the template record referenced by the FlowSet ID/template ID.
Padding	Should be inserted to align the end of the FlowSet on a 32 bit boundary. Pay attention that the Length field will include those padding bits.

When interpreting the NetFlow Version 9 data FlowSet format, note that the fields cannot be parsed without a corresponding template ID. If a data FlowSet that does not have an appropriate template ID is received, the record should be discarded.

Options Template Format

One additional record type is very important within the NetFlow Version 9 specification: an options template (and its corresponding options data record). Rather than supplying information about IP flows, options are used to supply “meta-data” about the NetFlow process itself. The format of the options template is detailed in Example B-1.

Example B-3 Options Template Format

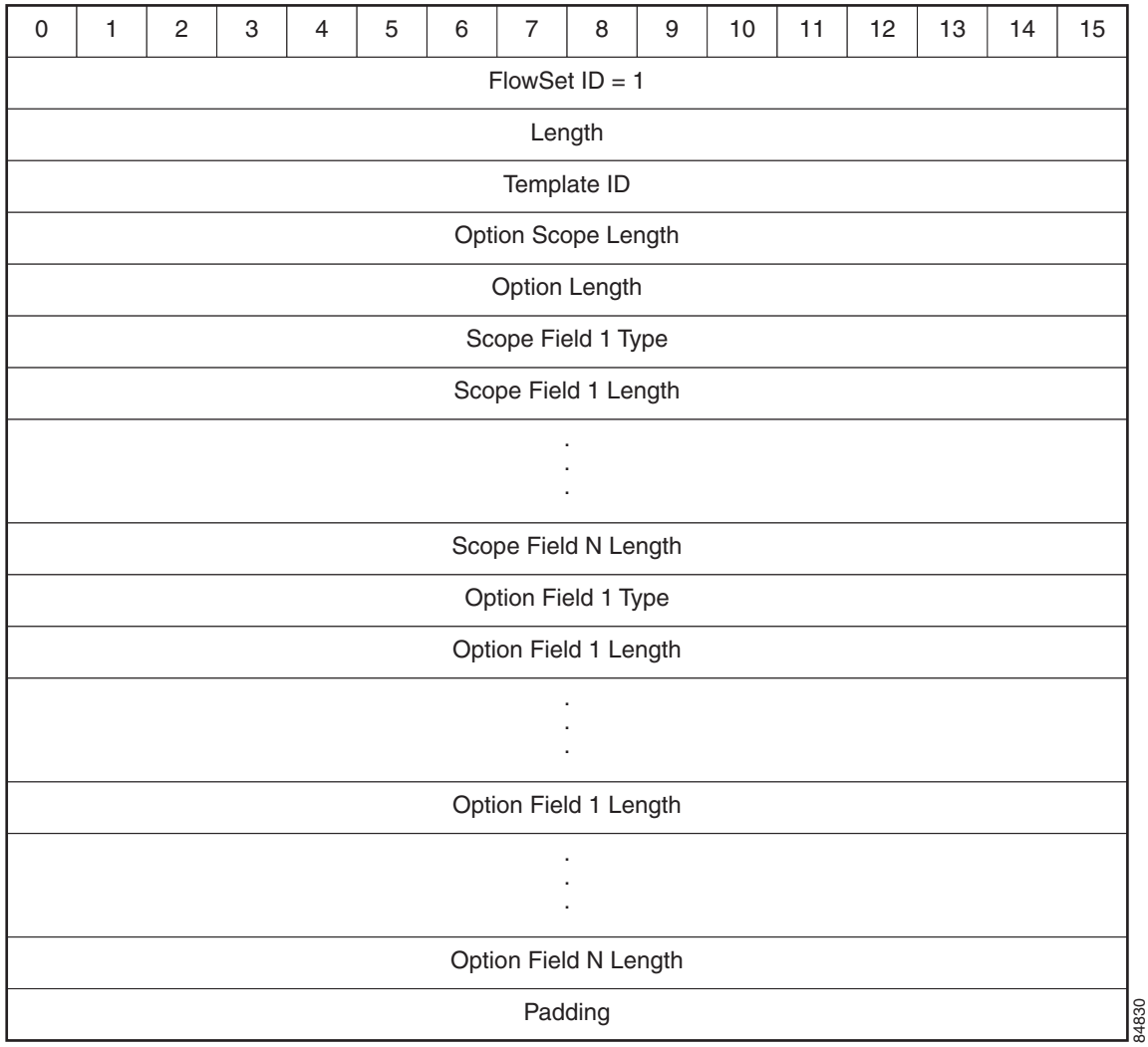


Table B-27 describes the Version 9 options template format.

Table B-27 Version 9 Options Template Format

Field Name	Description
FlowSet ID = 1	Used to distinguish template records from data records. A template record always has a FlowSet ID of 1. A data record always has a nonzero FlowSet ID which is greater than 255.
Length	The total length of this FlowSet. Because an individual template FlowSet can contain multiple template IDs, the length value should be used to determine the position of the next FlowSet record, which could be either a template or a data FlowSet. Length is expressed in TLV format, meaning that the value includes the bytes used for the FlowSet ID and the length bytes themselves, as well as the combined lengths of all template records included in this FlowSet.
Template ID	As a router generates different template FlowSets to match the type of NetFlow data it will be exporting, each template is given a unique ID. This uniqueness is local to the router that generated the template ID. The Template ID is greater than 255. Template IDs inferior to 255 are reserved.
Option Scope Length	The length in bytes of any scope fields contained in this options template (the use of scope is described below).
Options Length	The length (in bytes) of any Options field definitions contained in this options template.
Scope Field 1 Type	The relevant portion of the NetFlow process to which the options record refers. Currently defined values follow: • 0x0001 System • 0x0002 Interface • 0x0003 Line Card • 0x0004 NetFlow Cache • 0x0005 Template For example, sampled NetFlow can be implemented on a per-interface basis, so if the options record were reporting on how sampling is configured, the scope for the report would be 0x0002 (interface).
Scope Field 1 Length	The length (in bytes) of the Scope field, as it would appear in an options record.
Option Field 1 Type	Represents the type of the field that appears in the options record. Possible values are detailed in Table B-25.
Option Field 1 Length	The length (in bytes) of the field, as it would appear in an options record.
Padding	Should be inserted to align the end of the FlowSet on a 32 bit boundary. Pay attention that the Length field will include those padding bits.

CNS NetFlow Collection Engine Binary Data File Format

Although CNS NetFlow Collection Engine, Release 5.0 supports the binary data file format that is backwards-compatible with CNS NetFlow Collection Engine, Release 4.0.1 for all pre-defined aggregation schemes, support for binary output is being phased out starting with the 5.0 release. In addition, the binary file format is not supported for modified or newly-defined aggregation schemes. If you want to reduce the space consumed by output file, you should enable compression for ASCII output.

See the *Cisco CNS NetFlow Collection Engine, Release 4.0 Installation and User Guide* for a description of the CNS NetFlow Collection Engine, Release 4.0.1 binary data file format. You can find this guide using the following:

http://www.cisco.com/univercd/cc/td/doc/product/rtrmgmt/cns_nfc/rel_4_0/ins_user/binary.htm

Logging

CNS NetFlow Collection Engine, Release 5.0 uses Log4J from the Apache Foundation to perform logging functions. In general, all logs can be tuned to provide the level and amount of logging desired.

Configuration

All logging configurations come from files stored in the **NFC_DIR/config** directory. Table D-1 displays the log configuration file for each component of CNS NetFlow Collection Engine, Release 5.0.

Table D-1 Log Configuration Components

Component	Log Configuration File
NFC	NFC_DIR/config/nfc-log4j.properties
CNS/XML Interface	NFC_DIR/config/nfcxml-log4j.properties
Process Watcher	NFC_DIR/config/nfcpw-log4j.properties
BGP Peer	NFC_DIR/config/nfcbgp-log4j.properties
Report Engine	NFC_DIR/config/nfcre-log4j.properties
Web-based UI	NFC_DIR/config/nfcweb-log4j.properties

Two settings stored in these configuration files are **log filename** and **logging level**. To customize the **log filename**, change the line with **log4j.appender…File=<default filename>** in the appropriate configuration file.

For example, to change the path to the CNS NetFlow Collection Engine log, file you would change:

```
log4j.appender.nfcLog.File=${NFC_DIR}/logs/nfc${NFC_PROG}.log
```

to something like:

```
log4j.appender.nfcLog.File=/tmp/nfc.log
```

To customize the **logging level**, change the line with **log4j.logger…=INFO, …** in the appropriate configuration file. Valid levels are **FATAL**, **ERROR**, **WARN**, and **INFO**.

For example, to change the logging level of CNS NetFlow Collection Engine from **INFO** to **ERROR** you would change:

```
log4j.logger.com.cisco.nfc.collector=INFO, nfcLog
```

to:

```
log4j.logger.com.cisco.nfc.collector=ERROR, nfcLog
```

See **http://jakarta.apache.org/log4j** for more details on how these configuration files work.

CNS NetFlow Collection Engine CNS/XML Interface

This appendix describes the CNS NetFlow Collection Engine CNS/Xtensible Markup Language (XML) interface. The CNS/XML interface is the external configuration and control API for CNS NetFlow Collection Engine client applications. It allows clients to query and modify CNS NetFlow Collection Engine configuration and to start and stop the collector. The interface is accessible through the Cisco CNS Integration Bus and requests and responses are formatted in XML.

Configuration

By default the CNS/XML interface is configured for autostart with the Process Watcher. To start the CNS/XML interface, run **nfcollector start nfcxm** or **nfcollector start all**. The CNS/XML interface has a default configuration but you can customize the configuration in **$NFC_DIR/nfc-config.xml**.

Table E-1 describes the XML elements of a **<cns- xml-interface>** element that you can customize.

Table E-1 XML Configuration Elements

Element	Description	Default Value
subject	The CNS Integration Bus subject on which the CNS/XML interface will listen for collector-specific requests.	cisco.mgmt.nfc.${hostname}
broadcast-subject	The CNS Integration Bus subject on which the CNS/XML interface will listen for requests intended for all collectors. Currently, only **<listCollectors/>** is supported on the broadcast subject.	cisco.mgmt.nfc
md5-password	The password used to create a message digest for each XML request received. Requests are password protected with a digest. If the digest sent with the message does not equal the digest generated at the collector the request is denied with a status of **AUTH_ERROR**.	password
service	See CNS Integration Bus documentation for a description of this setting. This element has string content and no attributes.	No default

Table E-1 XML Configuration Elements (continued)

Element	Description	Default Value
network	See CNS Integration Bus documentation for a description of this setting. This element has string content and no attributes.	No default
daemon	See CNS Integration Bus documentation for a description of this setting. This element has string content and no attributes.	No default

To override the default settings, add a **<cns-xml-interface>** element to the **<nfc>** element in **NFC_DIR/config/nfc-config.xml**. (See **NFC_DIR/config/nfc-config.xsd** for the proper placement of the cns-xml-interface element.)

For example,

```
<cns-xml-interface>
  <subject>my.subject</subject>
  <broadcast-subject>my.broadcast.subject</broadcast-subject>
  <md5-password>mypassword</md5-password>
</cns-xml-interface>
```

Note If you customize the interface in **nfc-config.xml,** you must include values for subject, broadcast-subject, and md5-password.

Usage

The details of how to use the CNS Integration Bus are outside the scope of this document. This section describes the following:

- Message format
- Supported XML requests
- Response status

Message Format

Messages sent to the CNS/XML interface should include the information described in Table E-2.

Table E-2 CNS/XML Interface Message Attributes

Field or Attribute	Description
DATA	The XML request.

Table E-2 CNS/XML Interface Message Attributes (continued)

Field or Attribute	Description
MD5	The client-side digest generated from the digest password and XML request. This value is stored as a byte array.
Reply subject	Optional. If not specified, the CNS/XML interface will use the subject on which the message was received appended with **.response**.

Supported XML Requests

The following section lists the methods supported by the CNS/XML interface. See **NFC_DIR/config/nfc-methods.xsd** and **NFC_DIR/config/nfc-types.xsd** for details of the XML content of these requests and their responses.

- addFilter
- modifyFilter
- removeFilter
- listFilters
- getFilter
- addAggregationScheme
- modifyAggregationScheme
- removeAggregationScheme
- listAggregationSchemes
- getAggregationScheme
- addAggregator
- modifyAggregator
- removeAggregator
- listAggregators
- getAggregator
- listNDESourceGroups
- getNDESourceGroup
- listKeyBuilders
- getKeyBuilder
- listValueBuilders
- getValueBuilder
- listCollectors
- getCollectorConfiguration
- setCollectorConfiguration
- startCollector

- stopCollector
- listNDESources
- getApplicationStats

All methods except **<listCollectors>** should be sent to a specific collector using the appropriate CNS Integration Bus subject. See **NFC_DIR/config/nfc-methods.xsd** for a complete list of supported operations.

Response Status

When a response comes back from the CNS/XML interface is has an XML attribute named **status** to indicate whether or not the request succeeded. Table E-3 describes these status attributes.

Table E-3 Status Attributes

Status	Description
GET_RESPONSE	Response contains query results.
SET_RESPONSE	Request to change configuration succeeded.
GET_ERROR	Request to get configuration failed. Response element should contain **<error>**, **<warn>**, and/or **<info>** children to explain.
SET_ERROR	Request to change configuration failed. Response element should contain **<error>**, **<warn>**, or **<info>** children to explain.
AUTH_ERROR	Digest sent with message did not match digest generated by the CNS/XML interface.
DOWN_ERROR	Request for collector configuration failed because the collector is not running.
XML_ERROR	The XML syntax of the request is incorrect.

Sample Work Flow

This appendix contains a CNS NetFlow Collection Engine, Release 5.0 sample work flow to use as a reference.

Step 1 Start CNS NetFlow Collection Engine by entering:

: **/opt/CSCOnfc/nfcollector start all**

Step 2 Start web service:

/opt/CSCOnfc/nfcollector start web

Step 3 From an Internet Explorer window, open the URL:

: **http://<nfc-hostname>:8080/nfc**

Step 4 Log in as:

nfcuser/nfcuser

Step 5 Navigate to **Configuration > Aggregators**.

Step 6 Click **Add Aggregator** and add an aggregator through the GUI. Make sure there is NDE traffic arriving at the port to which the aggregator is listening.

Step 7 Wait until you see a new entry in the **filesready** file and view that data file. It should look like the following:

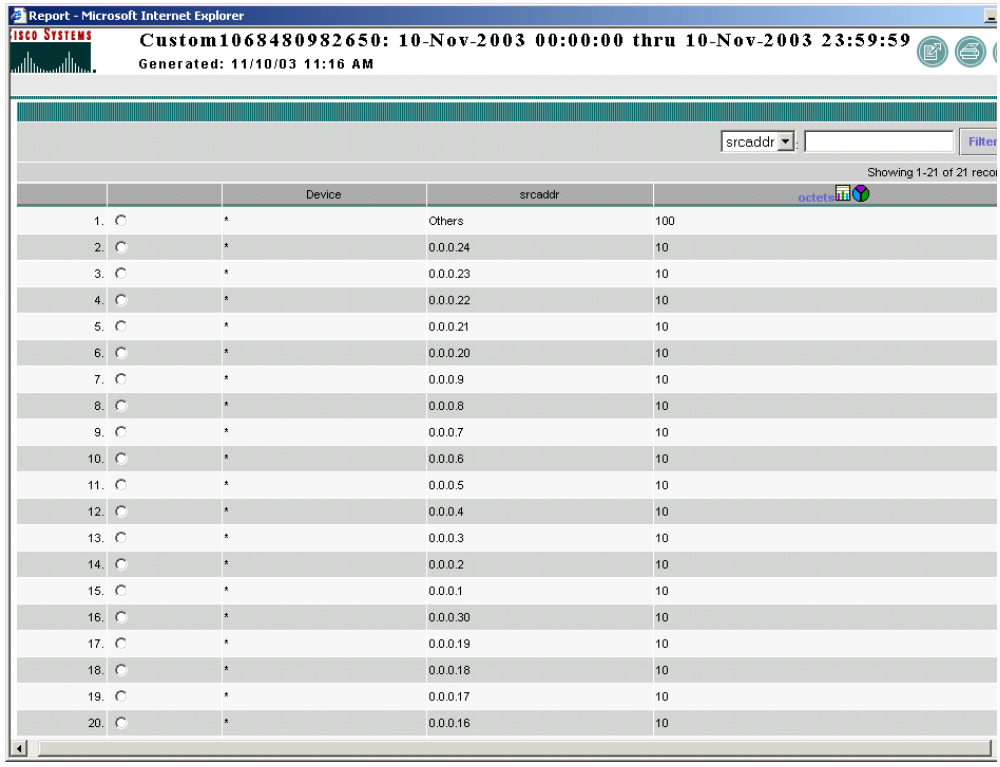
```
<?xml version="1.0"?>
<nfc-output-header xmlns="http://www.cisco.com/nfc/output" version="1.0">
    <source>64.102.41.45</source>
    <aggregator>DetailHostMatrixTest</aggregator>
    <aggregation-scheme name="DetailHostMatrix">
        <key name="srcaddr" type="ipaddress"/>
        <key name="dstaddr" type="ipaddress"/>
        <key name="srcport" type="string"/>
        <key name="dstport" type="string"/>
        <key name="protocol" type="string"/>
        <value name="pkts" type="integer"/>
        <value name="octets" type="integer"/>
        <value name="flows" type="integer"/>
        <value name="starttime" type="utc"/>
        <value name="endtime" type="utc"/>
    </aggregation-scheme>
    <delimiter>|</delimiter>
    <period-minutes>1</period-minutes>
    <starttime>1067636580</starttime>
    <endtime>1067636640</endtime>
    <flows>27</flows>
    <missed>0</missed>
```

```
    <records>27</records>
</nfc-output-header>
0.0.0.1|0.0.0.1|1|1|ICMP|1|1|1|1067636618|1067636625
0.0.0.2|0.0.0.2|2|2|ICMP|1|1|1|1067636618|1067636625
0.0.0.3|0.0.0.3|3|3|ICMP|1|1|1|1067636618|1067636625
0.0.0.4|0.0.0.4|4|4|ICMP|1|1|1|1067636618|1067636625
0.0.0.5|0.0.0.5|5|5|ICMP|1|1|1|1067636618|1067636625
0.0.0.6|0.0.0.6|6|6|ICMP|1|1|1|1067636618|1067636625
0.0.0.7|0.0.0.7|7|7|ICMP|1|1|1|1067636618|1067636625
0.0.0.8|0.0.0.8|8|8|ICMP|1|1|1|1067636618|1067636625
0.0.0.9|0.0.0.9|9|9|ICMP|1|1|1|1067636618|1067636625
0.0.0.10|0.0.0.10|10|10|ICMP|1|1|1|1067636618|1067636625
0.0.0.11|0.0.0.11|11|11|ICMP|1|1|1|1067636618|1067636625
0.0.0.12|0.0.0.12|12|12|ICMP|1|1|1|1067636618|1067636625
0.0.0.13|0.0.0.13|13|13|ICMP|1|1|1|1067636618|1067636625
0.0.0.14|0.0.0.14|14|14|ICMP|1|1|1|1067636618|1067636625
0.0.0.15|0.0.0.15|15|15|ICMP|1|1|1|1067636618|1067636625
0.0.0.16|0.0.0.16|16|16|ICMP|1|1|1|1067636618|1067636625
0.0.0.17|0.0.0.17|17|17|ICMP|1|1|1|1067636618|1067636625
0.0.0.18|0.0.0.18|18|18|ICMP|1|1|1|1067636618|1067636625
0.0.0.19|0.0.0.19|19|19|ICMP|1|1|1|1067636618|1067636625
0.0.0.20|0.0.0.20|20|20|ICMP|1|1|1|1067636618|1067636625
0.0.0.21|0.0.0.21|21|21|ICMP|1|1|1|1067636618|1067636625
0.0.0.22|0.0.0.22|22|22|ICMP|1|1|1|1067636618|1067636625
0.0.0.23|0.0.0.23|23|23|ICMP|1|1|1|1067636618|1067636625
0.0.0.24|0.0.0.24|24|24|ICMP|1|1|1|1067636618|1067636625
0.0.0.25|0.0.0.25|25|25|ICMP|1|1|1|1067636618|1067636625
0.0.0.26|0.0.0.26|26|26|ICMP|1|1|1|1067636618|1067636625
0.0.0.27|0.0.0.27|27|27|ICMP|1|1|1|1067636618|1067636625
```

Step 8 If no output is generated after you have waited for an entire aggregation period, check the **/opt/CSCOnfc/logs/nfc.log** to see if anything unexpected occurred.

Step 9 Go back to the web browser and navigate to **Reports > Common Reports > Today's Top Talkers**, generate a report on the data files produced by your aggregator.

Step 10 View the report, which shows the top 20 talkers based on CNS NetFlow Collection Engine data files of the current day. You should see a report screen like the following:



Step 11 Add a post processor by editing the configuration file **/opt/CSCOnfc/config/nfc- config.xml**. In the writer of your aggregator, add the DNS lookup post processor:

```
<writer>
    <ascii-writer>
        <use-compression>false</use-compression>
            <output-postprocessors>
        <output-postprocessor>/opt/CSCOnfc/bin/dnslookup.sh</output- postprocessor>
        </output-postprocessors>
    </ascii-writer>
</writer>
```

Step 12 Restart CNS NetFlow Collection Engine and wait for the next data file to be generated.

Step 13 Check the result. It should look like the following sample:

```
<?xml version="1.0"?>
<nfc-output-header xmlns="http://www.cisco.com/nfc/output" version="1.0">
    <source>64.102.41.45</source>
    <aggregator>DetailHostMatrixTest</aggregator>
    <aggregation-scheme name="DetailHostMatrix">
        <key name="srcaddr" type="ipaddress"/>
        <key name="dstaddr" type="ipaddress"/>
        <key name="srcport" type="string"/>
        <key name="dstport" type="string"/>
        <key name="protocol" type="string"/>
        <value name="pkts" type="integer"/>
        <value name="octets" type="integer"/>
        <value name="flows" type="integer"/>
        <value name="starttime" type="utc"/>
```

```
        <value name="endtime" type="utc"/>
    </aggregation-scheme>
    <delimiter>|</delimiter>
    <period-minutes>1</period-minutes>
    <starttime>1067636580</starttime>
    <endtime>1067636640</endtime>
    <flows>27</flows>
    <missed>0</missed>
    <records>27</records>
</nfc-output-header>
redeye.cisco.com|flashbulb.cisco.com|1|1|ICMP|1|1|1|1067636618|1067636625
redeye.cisco.com|lenscap.cisco.com|2|2|ICMP|1|1|1|1067636618|1067636625
…
```

CNS NetFlow Collection Engine Migration Tools

CNS NetFlow Collection Engine, Release 5.0.2 includes tools to facilitate migrating configurations and data from previous versions of CNS NetFlow Collection Engine.

This appendix includes the following sections:

- CNS NetFlow Collection Engine Configuration Migration, page G-1
- CNS NetFlow Collection Engine Data Migration, page G-2

CNS NetFlow Collection Engine Configuration Migration

CNS NetFlow Collection Engine, Release 5.0 includes a tool to facilitate migrating configurations from previous CNS NetFlow Collection Engine releases to the new XML format. The tool **migrateConfig.sh** is located under **NFC_DIR/tools**. When migrating a configuration, the tool ignores comments but assumes that blank lines separate definitions in **nfconfig.file**.

Using the Configuration Migration Tool

The command line syntax for this tool is:

```
$NFC_DIR/tools/migrateConfig.sh [-f <output-xml-file>] [-c <old-config-dir>]
```

where **<output-xml-file>** contains the migrated configuration and **<old-config-dir>** points to the directory containing old configuration files.

If command line arguments are not specified and the **NFC_DIR** environment variable is set:

- **NFC_DIR/config/nfc-config.xml** is used for **<output-xml-file>**
- **NFC_DIR/config** is used for **<old-config-dir>**

If command line arguments are not specified and the **NFC_DIR** environment variable is not set:

- **/opt/CSCOnfc/config/nfc-config.xml** is used for **<output-xml-file>**
- **/opt/CSCOnfc/config** is used for **<old-config-dir>**

If the **NFC_RESOURCEFILE** environment variable is not specified, the **migrateConfig.sh** tool looks for **<old-config-dir>/nf.resources**.

CNS NetFlow Collection Engine Data Migration

CNS NetFlow Collection Engine, Release 5.0 includes a tool to facilitate migrating data from previous CNS NetFlow Collection Engine releases to the new XML format that is required to generate reports. The tools **migrateDataFile.sh** and **migrateDataDir.sh** are located under **NFC_DIR/tools**. The **migrateDataFile.sh** tool converts the header in an CNS NetFlow Collection Engine output file to the new XML header format. The **migrateDataDir.sh** tool invokes **migrateDataFile.sh** for all files contained in a specified directory.

Using the Data Migration Tool

The command line syntax for the data migration tool is:

$NFC_DIR/tools/migrateDataFile.sh <file>

where **<file>** is an CNS NetFlow Collection Engine output file. If the file is not a text file or is not a valid CNS NetFlow Collection Engine output file, an error message is printed to standard error and a non-zero exit status is returned.

The tool **migrateDataDir.sh** is invoked with no arguments and prompts for the top-level directory containing data files such as **/opt/CSCOnfc/Data** to convert.

Note that the data migration tool only supports ASCII-format output files. Compressed output files must first be unzipped, and binary-format files must first be converted to ASCII. Also note that the cafeteria-style aggregation format in CNS NetFlow Collection Engine Release 4.0 is currently not supported.

A

active-time-value 4-12
Active Time value builder 2-15
address-range-map-key 4-5
Address Range Map key builder 2-9
Advanced configuration 2-24
aggregation scheme 4-15
Aggregation schemes 2-18
aggregator
  attributes 4-16
  creating 4-16
  elements 4-16
Aggregators 2-17

B

basic problems
  recommended actions A-5
BGP Attribute key builder 2-9
bgp-attr-key 4-6
bgp-attr-post-agg-key 4-6
bgp-complete-as-path-key 4-7
BGP Peer 2-23
bit-field-key 4-8
Bit Field key builder 2-10
Border Gateway Protocol (BGP) peer 5-8

C

CNS/XMLInterface
  supported requests E-3
CNS/XML interface
  configuration E-1
  message format E-2
CNS NetFlow Collection Engine
  architecture 1-5
  binary data file format C-1
  configuration 2-6
  configuration and resource files 4-1
  data file format 3-1
  Device and IOS Release Support 1-2
  functions 1-4, 1-5
  logging functions D-1
  overview 1-1
  overview illustration 1-4
  sample work flow F-1
  starting 2-1
Collector subsystem (NFCollector) 1-6
command conventions xi
compatibility
  IOS software 1-2
configuration elements 4-2
configuration migration tool G-1
conventions, command xi
creating an aggregator 4-16
customizing CNS NetFlow Collection Engine 4-1

D

data export
  compatibility matrix 1-2
  format 1-3
  mechanism 1-2
data file
  format 3-4

partial 3-8
data file directory 3-1
data file format C-1
data filenames 3-4
data migration tool G-2

E

end-time-value 4-12
End Time value builder 2-15
event service 5-3

F

fdcount utility A-2
fdget utility A-4
fdplayback utility A-4
Fields 2-7
fields 4-4
filesready log file 3-9
Filters 2-19
flow cache 1-2
flow-count-value 4-12
Flow Count value builder 2-15
flows
defined 1-1

G

get_bgp_rib utility A-3
global settings 2-6

I

inetaddress-key 4-8
Integer key builder 2-11
Integer Range Map key builder 2-11
interface-name-key 4-10
interface name support 5-10
interface settings 2-2
Interface SNMP Name key builder 2-10
interger-key 4-9
interger-range-map-key 4-9
Internet Address key builder 2-11
IP address
for configuration 1-4
IP packets 1-1

K

key builder
Address Range Map 2-9
address-range-map-key 4-5
BGP Attribute 2-9
bgp-attr-key 4-6
bgp-attr-post-agg-key 4-6
bgp-complete-as-path-key 4-7
bgp-complete-as-path-post-agg-key 4-7
Bit Field 2-10
bit-field-key 4-8
inetaddress-key 4-8
Integer 2-11
interface-name-key 4-10
Interface SNMP Name 2-10
interger-key 4-9
Interger Range Map 2-11
interger-range-map-key 4-9
Internet Address 2-11
masked-inetaddress-key 4-10
Masked Internet Address 2-12
mask-field-inetaddress-key 4-10
Mask Field Internet Address 2-12
Multi-Field Map 2-12
multi-field-map-ke 4-11
Option Data 2-13
option-data-key 4-11
String 2-14

string-key 4-11
key builder data types 4-13
key builders 2-8
keys and values 4-5

L

log file
filesready 3-9
log files 4-29
configuration from command line 4-30
logging configurations D-1
login screen 2-3
logs 2-39

M

masked-inetaddress-key 4-10
mask-field-inetaddress-key 4-10
Mask Field Internet Address key builder 2-12
max-burst-rate-value 4-12
Max Burst Rate value builder 2-16
Multi-Field Map editor 2-12
multi-field-map-key 4-11

N

ndget utility A-3
NetFlow data export
hardware supported 1-2
NetFlow export datagram formats B-1
NetFlow Export Source Access List 2-22
NetFlow Export Source Groups 2-21
NetFlow services
device and IOS release support 1-2
overview 1-1
nfc_bin_to_ascii converter utility A-5
nfc_gunzip utility A-5
nfcollector status command A-1

O

option-data-key 4-11
Option Data key builder 2-13
options data file
format 3-9

P

packets
IP 1-1
Port Statistics 2-38
Process Watcher 5-1

R

rate-value 4-13
Rate value builder 2-16
Report Generator 5-4
reports
Custom 2-26
Scheduled 2-30
rolling log files 4-29
rolloing file option 4-30
rollover schedule elements 4-31

S

Scheduled reports 2-30
showpacketlog utility A-3
show-tech command A-2
Sorting and Graphing 2-35
Source Statistics 2-38
start-time-value 4-13
Start Time value builder 2-16
Status 2-37

string-key 4-11
String key builder 2-14
sum-value 4-13
Sum value builder 2-16

T

traffic flows
  description 1-1
traffic statistics
  information types 1-3
troubleshooting 2-40

U

UDP
  exporting NetFlow data to port 1-4
  port number configuration 1-4
user interface
  Advanced 2-24
  Aggregation schemes 2-18
  Aggregators 2-17
  BGP Peer 2-23
  configuration 2-6
  Fields 2-7
  Filters 2-19
  global settings 2-6
  key builders 2-8
  logs 2-39
  navigation 2-4
  NetFlow Export Source Access List 2-22
  NetFlow Export Source Groups 2-21
  Reports 2-25
  statistics 2-38
  Status 2-37
  troubleshooting 2-40
  value builders 2-14

V

value builder
  Active Time 2-15
  active-time-value 4-12
  End Time 2-15
  end-time-value 4-12
  Flow Count 2-15
  flow-count-value 4-12
  Max Burst Rate 2-16
  max-burst-rate-value 4-12
  Rate 2-16
  rate-value 4-13
  Start Time 2-16
  start-time-value 4-13
  Sum 2-16
  sum-value 4-13
value builder data types 4-14
value builders 2-14
Version 1 NetFlow export datagram
  description 1-3
  Flow Record Format B-3
  flow record format B-3
  Header Format B-3
  header format B-3
Version 3 NetFlow export datagram
  Header Format B-3
  header format B-3
Version 5 NetFlow export datagram
  description 1-3
  Flow Record Format B-4
  flow record format B-4
Version 7 NetFlow export datagram
  description 1-3
  flow record format B-5
  Header Format B-5
  header format B-5
Version 8 NetFlow export datagram
  description 1-3

DestOnly flow record format B-12
FullFlow flow record format B-13
header format B-6
PrePortProtocol flow record format B-10
RouterAS flow record format B-6
RouterDstPrefix flow record B-7
RouterPrefix flow record format B-8
RouterProtoPort flow record B-7
RouterSrcPrefix flow record format B-8
SrcDst flow record format B-13
TosAS flow record format B-9
TosDstPrefix flow record format B-11
TosPrefix flow record format B-11
TosProtoPort flow record format B-9
TosSrcPrefix flow record format B-11
Version 9 NetFlow export datagram
Data FlowSet Format B-27
Data FlowSet format B-24
description 1-3
Export Packet B-14
field types B-17
Options Template Format B-25, B-26
packet header format B-15
Template FlowSet format B-17
template FlowSet format B-15

W

web-based UI
troubleshooting tips A-9

X

XML
header format 3-7
XML Schema 4-1