



Cisco NetFlow Collector User Guide

Release 6.0

Americas 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: 408 527-0883

Customer Order Number:
Text Part Number: OL-11399-01

THE SPECIFICATIONS AND INFORMATION REGARDING THE PRODUCTS IN THIS MANUAL ARE SUBJECT TO CHANGE WITHOUT NOTICE. ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS MANUAL ARE BELIEVED TO BE ACCURATE BUT ARE PRESENTED WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. USERS MUST TAKE FULL RESPONSIBILITY FOR THEIR APPLICATION OF ANY PRODUCTS.

THE SOFTWARE LICENSE AND LIMITED WARRANTY FOR THE ACCOMPANYING PRODUCT ARE SET FORTH IN THE INFORMATION PACKET THAT SHIPPED WITH THE PRODUCT AND ARE INCORPORATED HEREIN BY THIS REFERENCE. IF YOU ARE UNABLE TO LOCATE THE SOFTWARE LICENSE OR LIMITED WARRANTY, CONTACT YOUR CISCO REPRESENTATIVE FOR A COPY.

The Cisco implementation of TCP header compression is an adaptation of a program developed by the University of California, Berkeley (UCB) as part of UCB's public domain version of the UNIX operating system. All rights reserved. Copyright © 1981, Regents of the University of California.

NOTWITHSTANDING ANY OTHER WARRANTY HEREIN, ALL DOCUMENT FILES AND SOFTWARE OF THESE SUPPLIERS ARE PROVIDED "AS IS" WITH ALL FAULTS. CISCO AND THE ABOVE-NAMED SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THOSE OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

IN NO EVENT SHALL CISCO OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF CISCO OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

CCVP, the Cisco logo, and Welcome to the Human Network are trademarks of Cisco Systems, Inc.; Changing the Way We Work, Live, Play, and Learn is a service mark of Cisco Systems, Inc.; and Access Registrar, Aironet, BPX, Catalyst, CCDA, CCDP, CCIE, CCIP, CCNA, CCNP, CCSP, Cisco, the Cisco Certified Internetwork Expert logo, Cisco IOS, Cisco Press, Cisco Systems, Cisco Systems Capital, the Cisco Systems logo, Cisco Unity, Enterprise/Solver, EtherChannel, EtherFast, EtherSwitch, Fast Step, Follow Me Browsing, FormShare, GigaDrive, HomeLink, Internet Quotient, IOS, iPhone, IP/TV, iQ Expertise, the iQ logo, iQ Net Readiness Scorecard, iQuick Study, LightStream, Linksys, MeetingPlace, MGX, Networkers, Networking Academy, Network Registrar, PIX, ProConnect, ScriptShare, SMARTnet, StackWise, The Fastest Way to Increase Your Internet Quotient, and TransPath are registered trademarks of Cisco Systems, Inc. and/or its affiliates in the United States and certain other countries.

All other trademarks mentioned in this document or Website are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (0710R)

Any Internet Protocol (IP) addresses used in this document are not intended to be actual addresses. Any examples, command display output, and figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses in illustrative content is unintentional and coincidental.

Cisco NetFlow Collector User Guide

© 2007 Cisco Systems, Inc. All rights reserved.



CONTENTS

About This Guide vii

- Objective vii
- Audience vii
- How This Guide Is Organized viii
- Command Syntax Conventions viii
- Obtaining Documentation, Obtaining Support, and Security Guidelines ix

CHAPTER 1-1

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 Cisco NetFlow Collector? 1-4
- Cisco NetFlow Collector Architectural Overview 1-5
 - Collector 1-6
 - Web-Based User Interface 1-6
 - Report Generator 1-6
 - BGP Peer 1-6

CHAPTER 2-1

Using the NetFlow Collector User Interface 2-1

- Starting the Cisco NetFlow Collector User Interface 2-1
- Customizing the Cisco NetFlow Collector Interface 2-2
- Using the Cisco NetFlow Collector User Interface 2-3
 - The NFC Login Window 2-3
 - Navigation 2-4
- Configuration 2-5
 - Aggregators 2-7
 - Adding Aggregators 2-7
 - Editing an Aggregator 2-8
 - Thresholds 2-9
 - Fields 2-10

Key Builders 2-11
BGP Attribute 2-13
Bit Field 2-14
Boolean 2-14
Byte Array 2-14
Customer Name 2-15
Egress PE 2-15
Ingress CE 2-16
Integer 2-16
Integer Range Map 2-17
Interface SNMP Name 2-17
IP Address 2-17
IP Address Range Map 2-18
Mac Address 2-18
Masked IP Address 2-18
Multi-Field Map 2-19
Option Data 2-20
Site Name 2-20
String 2-21
Subnet Address 2-21
Value Builders 2-21
Active Time 2-23
Directional Sum 2-23
End Time 2-23
Flow Count 2-23
Max Flow Byte Rate 2-24
Rate 2-24
Start Time 2-24
Sum 2-24
Sum with Sampling Estimation 2-25
Aggregation Schemes 2-25
Filters 2-26
NetFlow Export Source Groups 2-27
NetFlow Export Source Access List 2-28
BGP Peer 2-29
Global 2-30
Advanced 2-30
Reports 2-31
Custom Reports 2-32
Report Templates 2-36

Scheduled Reports 2-37
Configuring Scheduled Reports 2-37
Displaying Scheduled Reports 2-41
Reporting Features 2-42
Sorting and Graphing 2-43
Trending 2-44
Export and Print 2-45
Filter 2-45
Drill Down 2-45
Status 2-45
Control 2-46
Statistics 2-46
Health Monitor Statistics 2-46
Port Statistics 2-47
Source Statistics 2-48
Logs 2-49

CHAPTER 3-1

Understanding the NetFlow Collector 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
Threshold Crossing Record File 3-10
Using the filesready File to Track Data Files 3-10
Options Data Filesready File 3-10
Threshold Filesready File 3-11

CHAPTER 4-1

Customizing the Cisco NetFlow Collector 4-1

Configuration and Resource Files 4-1
XML Configuration and Schema 4-1
Data Collection and Aggregation 4-2
Overview of NetFlow Collector Release 6.0 Configuration Information 4-3
Fields 4-4

Keys and Values 4-4
Key Builder Types 4-5
Value Builder Types 4-13
Key and Value Builder Data Types 4-15
Creating an Aggregation Scheme 4-17
Creating an Aggregator 4-18
Creating a Filter 4-21
Creating a Map 4-22
Creating a Multi-Field Map 4-23
Creating an Option Data Map 4-24
Creating Source Groups 4-26
Creating Access Lists 4-26
Creating a Threshold 4-26
Global Settings 4-28
Setting the Time Zone 4-29
Memory Usage 4-29
Disk Space 4-31
Filters 4-31
Aggregation 4-31
Data File and Disk Space Options 4-31
Monitoring Disk Usage 4-32
Process Watcher 4-32
Configuration 4-33
Event Service 4-34
Configuration 4-34
Scheduled Reports 4-35
BGP Peer 4-38
Starting and Stopping the BGP Peer 4-39
Configuration 4-39
Interface Name Support 4-40
Packet Log 4-41
Site In/Out Traffic Summary 4-42

APPENDIX A-1

Troubleshooting the Cisco NetFlow Collector A-1

Using the nfcollector list Command A-1
Using the show-tech Command to Capture Troubleshooting Information A-2
NetFlow Collector Tools and Utilities A-2
fdcount Utility A-2

ndeget Utility A-3
get_bgp_rib Utility A-3
fdget Utility A-3
fdplayback Utility A-4

Solving NetFlow Collector Problems A-4

APPENDIX B-1**Logging B-1**

Configuration B-1

Configuring the Logger from the Command Line B-2

Rolling File Option for NFC Logs Files B-3

Daily Rolling File Option for NFC Log Files B-3

APPENDIX C-1**Cisco NetFlow Collector Sample Work Flow C-1**

APPENDIX D-1**NetFlow Fanout D-1**

Command Line Syntax D-1

Configuration Note D-2

APPENDIX E-1**Cisco NetFlow Collector Binary Data File Format E-1**

APPENDIX F-1**Cisco NetFlow Collector CNS/XML Interface F-1**



About This Guide

Objective

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

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. Cisco NetFlow Collector provides fast, scalable, and economical data collection from multiple export devices exporting NetFlow data records.

Cisco NetFlow Collector, Release 6.0 introduces a tiered netflow collection architecture that provides increased scalability and performance. The role of the first tier (Tier 1) maps to the NFC functionality of Cisco NetFlow Collector 5.0.3 with the addition of new features described in *Release Notes for Cisco NetFlow Collector, Release 6.0*.

Cisco NetFlow Collector, Release 6.0 supports new Cisco NetFlow Collector Tier 2 functionality, also referred to as Multi NetFlow Collector. The Multi NetFlow Collector runs on separate server hardware and provides an aggregation layer that correlates data from several Tier 1 instances.

Prior to reading this guide, you should read the *Release Notes for Cisco NetFlow Collector, Release 6.0* document. These release notes provide information about known software and documentation problems and any last minute information about the NetFlow Collector software not available when this guide was produced.

In previous releases, this product was referred to as Cisco NetFlow Collection Engine (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, Red Hat Enterprise Linux, or Sun Microsystems's Solaris Operating System.

How This Guide Is Organized

This guide is organized as follows:

[Chapter 1, “Overview,”](#) describes the Cisco NetFlow Collector application.

[Chapter 2, “Installing the Cisco NetFlow Collector,”](#) describes how to install the Cisco NetFlow Collector application.

[Chapter 3, “Configuring the Cisco NetFlow Collector,”](#) describes how to configure the Cisco NetFlow Collector and then validate that it is operating properly.

[Chapter 4, “Customizing the Cisco NetFlow Collector,”](#) describes how to customize the NetFlow Collector operations.

[Appendix A, “Troubleshooting the Cisco NetFlow Collector,”](#) contains troubleshooting information in case you encounter problems while using the Cisco NetFlow Collector.

[Appendix B, “Logging,”](#) describes how to configure the NetFlow Collector logging functions.

[Appendix C, “Cisco NetFlow Collector Sample Work Flow,”](#) contains a sample work flow to use as a reference.

[Appendix D, “NetFlow Fanout,”](#) describes the NetFlow Collector flow-fanout tool.

Appendix E and F explain functionality that was present in past releases of NetFlow Collector that have been deprecated in NetFlow Collector Release 6.

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.
<i>italic</i>	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 <i>Control</i> . For example, when you read ^D or <i>Ctrl-D</i> , 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, Obtaining Support, and Security Guidelines

For information on obtaining documentation, obtaining support, providing documentation feedback, security guidelines, and also recommended aliases and general Cisco documents, see the monthly *What's New in Cisco Product Documentation*, which also lists all new and revised Cisco technical documentation, at:

<http://www.cisco.com/en/US/docs/general/whatsnew/whatsnew.html>



CHAPTER 1

Overview

This chapter describes the Cisco NetFlow Collector (NFC) application, which is used with the NetFlow services data export feature on Cisco routers and Catalyst switches.

This chapter includes the following sections:

- [What Are NetFlow Services?](#)
- [What Is Cisco NetFlow Collector?](#)
- [Cisco NetFlow Collector 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) and Stream Control Transmission Protocol (SCTP) datagrams, which NetFlow Collector 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 timer expires.

For flows that remain continuously active, flow cache entries expire after a specified period of time, for example 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 NetFlow Collector, 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,”](#) in the *Cisco NetFlow Collector User Guide* 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). NetFlow Collector 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 NetFlow Collector. For more information on the distinctions among the NetFlow data export formats, see [Appendix B, “NetFlow Export Datagram Formats,”](#) in the *Cisco NetFlow Collector User Guide*.

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).

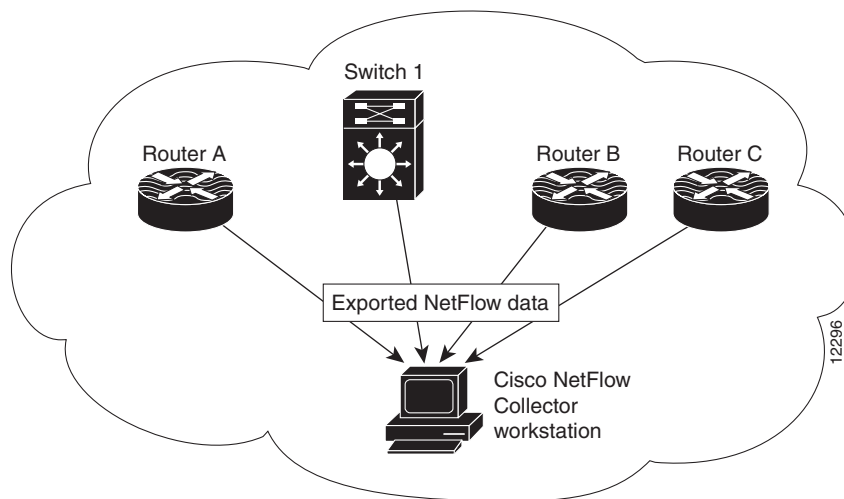
**Caution**

Throughout this publication there are numerous examples of NetFlow Collector 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 Cisco NetFlow Collector?

The Cisco NetFlow Collector application 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 NetFlow Collector UDP and SCTP ports.

Figure 1-1 *NetFlow Collector 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 or SCTP port number (a logical port designator) that identify NetFlow Collector as the receiver of flows from this export device. The port number is a user-configurable designator: you can configure NetFlow Collector to listen for flows on a number of different ports, and then configure your export devices so that each device exports flows to a dedicated port, or have a number of devices export flows to the same, shared port.

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

Cisco NetFlow Collector 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

Cisco NetFlow Collector collects and summarizes (aggregates) data into data files based on user-defined criteria specified in a NetFlow Collector *aggregator*. An *aggregator* is an aggregation task defined by a set of user-configurable attributes that specify how NetFlow Collector summarizes the traffic flows that are received. Three 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 (summarized).
- Port - UDP or SCTP destination port configured on the export device.

Cisco NetFlow Collector 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 NetFlow Collector for your operating context. Moreover, starting 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.

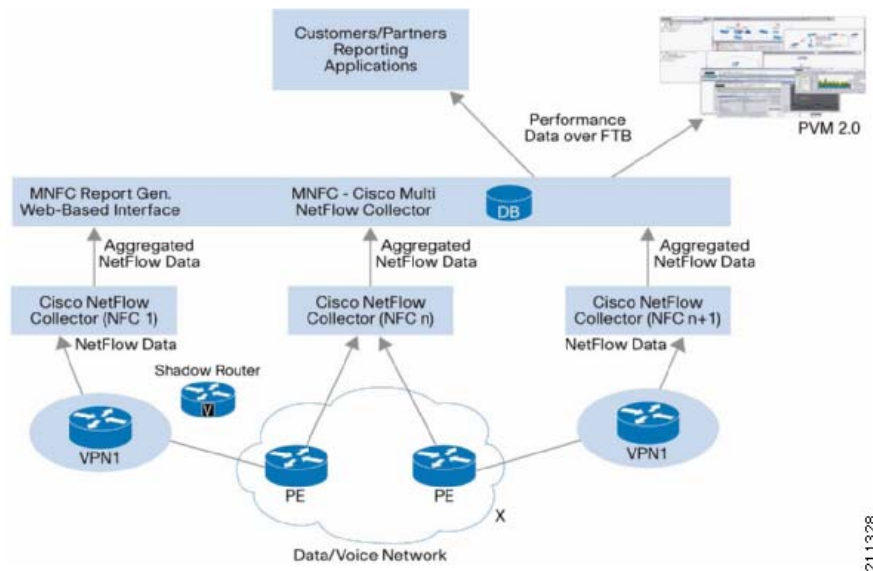
For more information about threads, aggregation schemes, and filters, see [Chapter 4, “Customizing the CNS NetFlow Collection Engine,”](#) in the *Cisco NetFlow Collector User Guide*.

Cisco NetFlow Collector Architectural Overview

Cisco NetFlow Collector consists of the following components:

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

These subsystems work together to provide Cisco NetFlow Collector functionality, including data collection, the user interface, configuration and control, and reporting. They also allow custom client applications to interface with Cisco NetFlow Collector. See [Figure 1-2](#) for a graphical representation of the Cisco NetFlow Collector system architecture.

Figure 1-2 NetFlow Collector System Architecture

Collector

The Collector subsystem collects NetFlow data, aggregates or summarizes 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.

Report Generator

The Report Generator produces on-demand, 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 Cisco NetFlow Collector output with BGP attributes.



CHAPTER 2

Using the NetFlow Collector User Interface

Cisco NetFlow Collector (NFC), Release 6.0 has a 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 Cisco NetFlow Collector User Interface, page 2-1](#)
- [Customizing the Cisco NetFlow Collector Interface, page 2-2](#)
- [Using the Cisco NetFlow Collector User Interface, page 2-3](#)
- [Configuration, page 2-5](#)
- [Reports, page 2-31](#)
- [Status, page 2-45](#)

Starting the Cisco NetFlow Collector User Interface

To start the Cisco NetFlow Collector User Interface, do the following.



Note

The Cisco NetFlow Collector User Interface requires JRE 1.5 or higher. You can download a plug-in for Java 1.5 or higher from java.sun.com, section **Downloads**, **J2SE** folder; and install it on the platform on which the browser will run.

Step 1 To run Cisco NetFlow Collector, 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 enter:

```
//<nfc-hostname>:8080/nfc
```

**Note**

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

Customizing the Cisco NetFlow Collector Interface

The NFC application includes the tool `/opt/CSCOnfc/bin/webconfig.sh` for configuring HTTP or HTTPS and the port number for accessing the web UI.

For example, to enable HTTPS access, do the following:

Step 1 To run the tool, enter the following:

```
/opt/CSCOnfc/bin/webconfig.sh
```

Step 2 You are prompted to configure HTTP or HTTPS access to the NFC web server.

Configure http or https access to the NFC web server:

```
[1] Access the NFC web server with http (unencrypted)
```

```
[2] Access the NFC web server with https (encrypted)
```

Select one:

Step 3 To select HTTPS, enter **2**.

Step 4 Enter the port number for web access.

```
Enter port number for web access [8443]
```

Step 5 Enter the keystore and certificate password. It must be at least 6 characters.

Step 6 Select a certificate type.

Certificate type:

```
[1] Create a self-signed certificate
```

```
[2] Import an existing certificate
```

Select one:

If you select 1, the window displays:

```
Creating keystore with self-signed certificate
```

```
Enter certificate validity period in days: [3650]
```

The subject name in the certificate is based on the hostname of this device by default. If the URL used to access NFC on this host contains a different name e.g. IP address, the browser will report a site name mismatch.

Step 7 Enter the subject hostname or IP address.

Step 8 When the web configuration is complete, the following is displayed:
NFC web configuration has been updated.

Table 2-1 describes additional settings that can be customized for the Cisco NetFlow Collector web-based UI.

Table 2-1 Cisco NetFlow Collector User Interface Settings

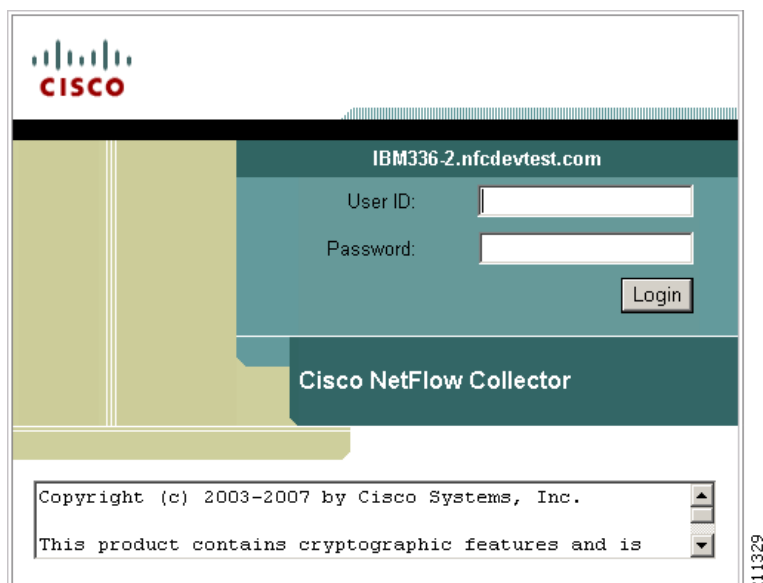
Setting	Description	Default Value	File
intfc- password	Digest password for the CNS/XML interface. Stored as a parameter to the InitServlet in the servlet configuration file. This setting must match the md5-password value of the CNS/XML interface.	password	NFC_DIR/tomcat/webapps/nfc/WEB-INF/web.xml
session-timeout	A session is started after 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

Using the Cisco NetFlow Collector User Interface

The following sections describe using the Cisco NetFlow Collector User Interface.

The NFC Login Window

When starting the Cisco NetFlow Collector, the first window that appears is the NFC login window, 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.

Figure 2-1 Cisco NetFlow Collector User Interface Login Window

To log in to Cisco NetFlow Collector, do the following:

Step 1 From the Login window, enter your User ID and Password.

Step 2 Click **Login**.

The Cisco NetFlow Collector Main window appears. From this window, you can select from the following tabs:

- Configuration
- Reports
- Status

See the following sections for information on these functions.

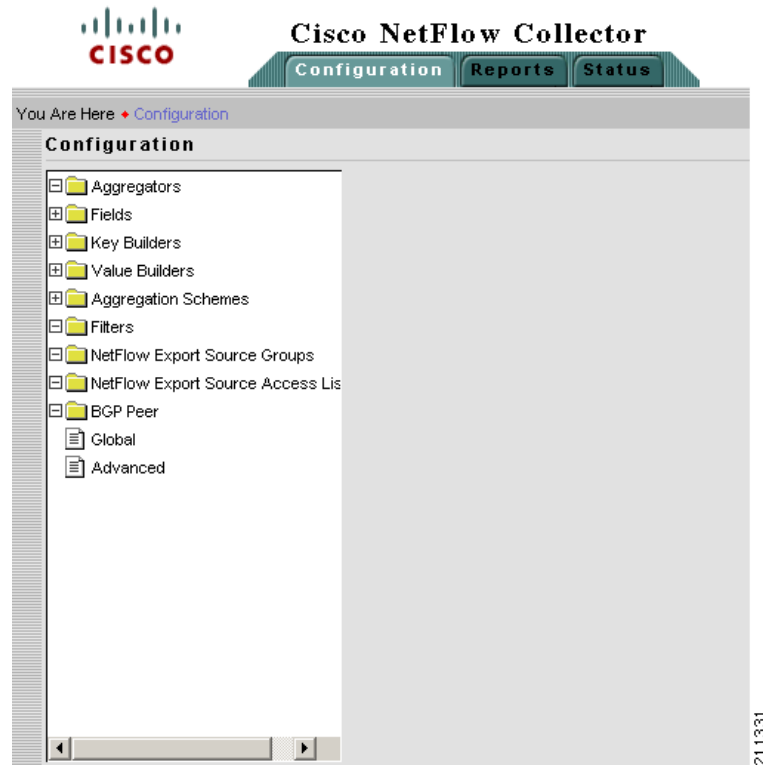
Navigation

You can move around the NFC web-based user interface (UI) from two levels. Across the top of all NFC windows are the NFC UI navigation tabs. These tabs are the first level of navigation in to the NFC UI, as shown in [Figure 2-2](#). From here you can select the **Configuration**, **Reports**, and **Status** tabs. The toolbar at the far right includes links to **Logout**, **Help**, and **About** windows.

Figure 2-2 NFC UI Navigation Tabs

Each section of NFC 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.

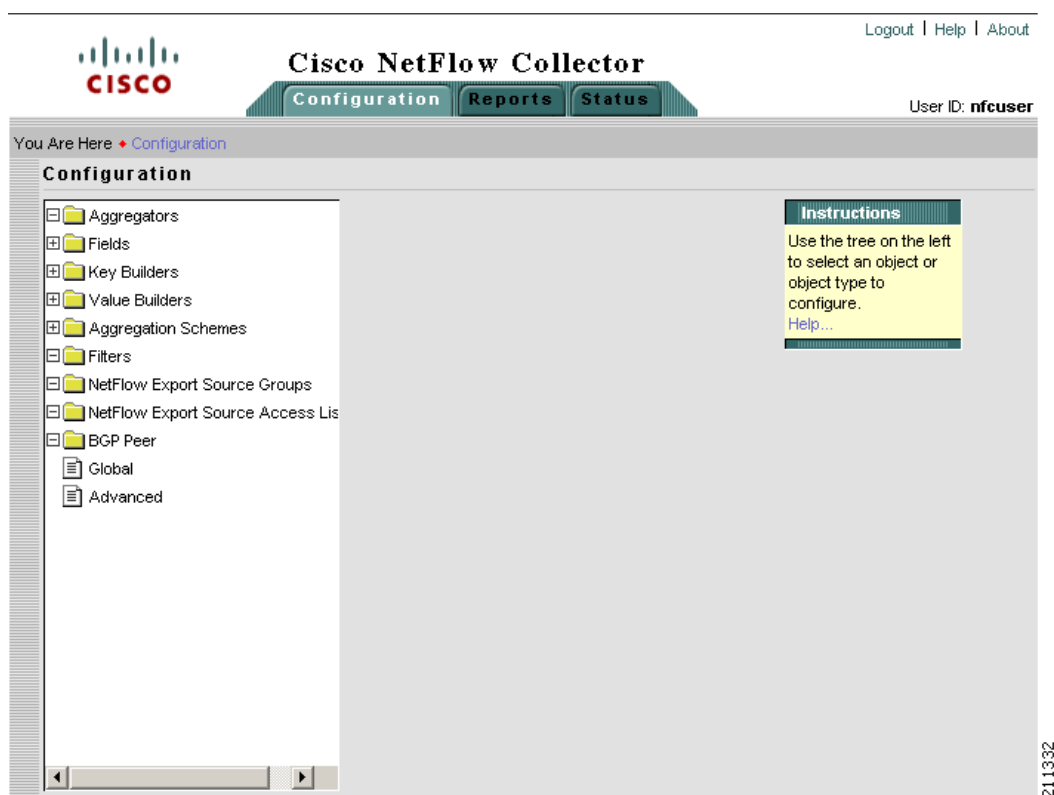
Figure 2-3 NFC UI Navigation Tree



Configuration

From the Configuration window you can perform tasks including specify global parameters; define fields, key builders, value builders and aggregators; and create filters.

From the Cisco NetFlow Collector **Main** window, click the **Configuration** tab. The Configuration window appears, as shown in [Figure 2-4](#).

Figure 2-4 NFC Configuration Window

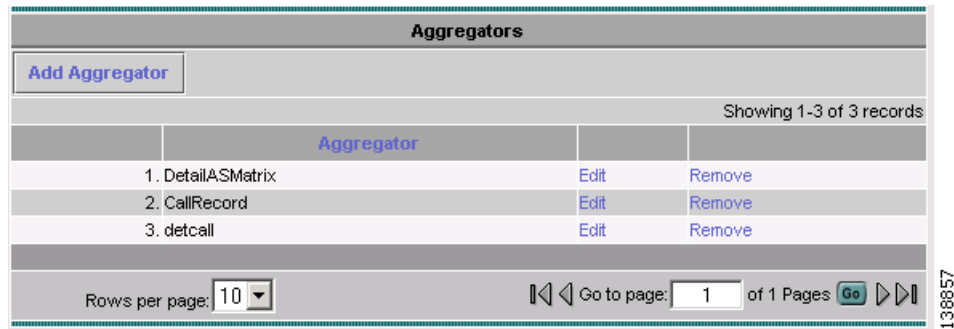
From this window you can access or configure the following:

- [Aggregators, page 2-7](#)
- [Fields, page 2-10](#)
- [Key Builders, page 2-11](#)
- [Value Builders, page 2-21](#)
- [Aggregation Schemes, page 2-25](#)
- [Filters, page 2-26](#)
- [NetFlow Export Source Groups, page 2-27](#)
- [NetFlow Export Source Access List, page 2-28](#)
- [BGP Peer, page 2-29](#)
- [Global, page 2-30](#)
- [Advanced, page 2-30](#)

Aggregators

Aggregators define how the Cisco NetFlow Collector receives NetFlow data, aggregates or combines the data, and generates output files. Click on the **Aggregators** folder of the NFC UI navigation tree to display a table of all existing aggregators, as shown in [Figure 2-5](#).

Figure 2-5 **Aggregators Window**



The screenshot shows the 'Aggregators' window. At the top is a header bar with the title 'Aggregators'. Below the header is a button labeled 'Add Aggregator'. Underneath the button, it says 'Showing 1-3 of 3 records'. The main content is a table with the following data:

Aggregator			
1. DetailASMatrix	Edit	Remove	
2. CallRecord	Edit	Remove	
3. detcall	Edit	Remove	

At the bottom of the window, there are pagination controls: 'Rows per page: 10' (with a dropdown arrow), 'Go to page: 1 of 1 Pages', a 'Go' button, and navigation arrows. A vertical text '138857' is visible on the right side of the window.

Adding Aggregators

From the Aggregators window, click on **Add Aggregator** to bring up the Add Aggregator window to define a new aggregator. See [Figure 2-6](#).

Figure 2-6 Add Aggregator Window

Add Aggregator			
Aggregator ID:	<input type="text"/>		
Aggregation Scheme:	ASHostMatrix		
Aggregation Period (mins):	1		
Port Number:	9991		
Protocol:	udp		
State:	active		
Data Set Path:	\${NFC_DIR}/Data		
Output Format:	default		
Compression:	<input type="checkbox"/>		
Maximum Disk Usage (MBs):	0		
Filter:	-		
Sort Output:	<input type="checkbox"/>		
Threshold Directory:	\${NFC_DIR}/threshold-d		
Threshold Output Format:	mixed		
Thresholds:	<input type="button" value="Add Threshold"/>		
	Showing 0-0 of 0 records		
	<table border="1"> <thead> <tr> <th>Threshold</th> </tr> </thead> <tbody> <tr> <td>No records.</td> </tr> </tbody> </table>	Threshold	No records.
	Threshold		
No records.			
Rows per page: 10 Go to page: 1 of 1 Pages <input type="button" value="Go"/>			
<input type="button" value="Submit"/>			

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

Editing an Aggregator

To modify or remove an existing aggregator, click **Edit** for the aggregator which you wish to modify or remove from the list of aggregators displayed in the Aggregator window (Figure 2-6). The **Modify Aggregator** window displays, as shown in Figure 2-7.

Figure 2-7 *Modify Aggregator Window*

Modify Aggregator			
Aggregator ID:	CallRec		
Aggregation Scheme:	HostMatrix		
Aggregation Period (mins):	<input type="text" value="5"/>		
Port Number:	<input type="text" value="2100"/>		
Protocol:	<input type="text" value="udp"/>		
State:	<input type="text" value="active"/>		
Data Set Path:	<input type="text" value="\${NFC_DIR}/Data"/>		
Output Format:	<input type="text" value="default"/>		
Compression:	<input type="checkbox"/>		
Maximum Disk Usage (MBs):	<input type="text" value="0"/>		
Filter:	<input type="text" value="--"/>		
Sort Output:	<input checked="" type="checkbox"/>		
Threshold Directory:	<input type="text" value="\${NFC_DIR}/threshold-c"/>		
Threshold Output Format:	<input type="text" value="mixed"/>		
Thresholds:	Add Threshold		
	Showing 1-2 of 2 records		
		Threshold	
	1. threshold1		Edit Remove
	2. threshold2		Edit Remove
	Rows per page:	<input type="text" value="10"/>	Go to page: <input type="text" value="1"/> of 1 Pages Go
			Modify Remove

To modify the selected aggregator, fill in the fields and click **Modify** to complete the operation. To remove the selected aggregator, click **Remove**.

**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.

Thresholds

Thresholds provide a way to generate events when values in the NetFlow Collector output cross a specified target value. You configure thresholds for each aggregator. A list of thresholds for an aggregator is displayed in the Add Aggregator window.

From the Add Aggregator window, click **Add Threshold** to add a new threshold. Click on the appropriate link in the threshold list to modify or remove an existing Threshold.

When adding and editing thresholds the windows are identical with the exception that you cannot change the threshold ID when modifying a threshold. Use this window to add, remove, and order threshold conditions.

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

The top item in the tree is the name of the threshold. Directly beneath this is a top-level threshold condition or expression. Add the top-level threshold condition or expression by selecting **Add condition** or **Add expression** when the top item is selected. If the top-level threshold condition or expression evaluates to true when the threshold is evaluated, a threshold-crossing log is created. See the [“Creating a Threshold” section on page 4-26](#) for more information about thresholds.

A threshold expression contains two or more expressions or conditions. Arbitrarily complex threshold evaluation logic can be specified in this way.

When creating a threshold condition, specify:

- Whether the comparison is greater than, less than, equals, or not-equals
- Which key or value is compared

Directly beneath the threshold condition is one or more value or range items. These determine the set of target values to which the comparison is applied. Add a value or range to the threshold condition by selecting Value or Range. For an integer condition, only integer values and ranges can be entered; only IP address values can be entered for address conditions.

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

To create an expression, specify the logical operator and, or, not-and, or 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 closer 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 because no cut, paste, or undo capability is provided. Changes are not committed until you select **Update Threshold** or **Remove Threshold**.

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 threshold can be added or updated.

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 NFC UI navigation tree displays a table of currently defined fields as shown in [Figure 2-8](#). Click **Edit** to modify a specific field, or **Remove** to remove a selected field. Click **Add Field** to bring up an empty form for defining a new field.

Aliases, alternate names for fields, are also shown in the navigation tree and table and can be added when a field is defined or modified

Figure 2-8 Fields Window

Fields			
Add Field			
Showing 1-10 of 345 records			
	Field	Numeric ID	
1.	BGP_IPV4_NEXT_HOP	18	Edit Remove
2.	bgpDestinationAsNumber	17	Edit Remove
3.	bgpNextAdjacentAsNumber	128	Edit Remove
4.	bgpNextHopIPv4Address	18	Edit Remove
5.	bgpNextHopIPv6Address	63	Edit Remove
6.	bgpPrevAdjacentAsNumber	129	Edit Remove
7.	bgpSourceAsNumber	16	Edit Remove
8.	BPG_IPV6_NEXT_HOP	63	Edit Remove
9.	CLASS_ID	51	Edit Remove
10.	classid	51	Edit Remove
Rows per page: <input type="text" value="10"/> Go to page: <input type="text" value="1"/> of 35 Pages Go			

The NetFlow Export Field window, [Figure 2-9](#), is displayed when adding or modifying a field. Fill in the form and click **Add** or **Modify** to complete the operation. From the Modify window you can also remove the currently displayed field. Click **Add Alias** or **Remove Alias** to add or remove an alias (alternate name) for this field. See the “Fields” section on page 4-4 for additional information about field definitions.

Figure 2-9 NetFlow Export Field Window

NetFlow Export Field	
Numeric ID: *	<input type="text"/>
Name: *	<input type="text"/>
Type:	<input type="text" value="Byte Array"/>
Aliases	
Add Alias	Remove Alias
Showing 0-0 of 0 records	
	Alias
No records.	
Rows per page: <input type="text" value="10"/> Go to page: <input type="text" value="1"/> of 1 Pages Go	
Add	

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 NFC UI navigation tree displays a table of currently defined key builders as shown in [Figure 2-10](#). Click **Edit** to modify a specific key builder, or **Remove** to remove a selected key builder. Click **Add Key Builder** to bring up an empty form for defining a new key builder.

Figure 2-10 Key Builders Window

Key Builders			
Add Key Builder			
Showing 1-10 of 29 records			
	Key Builder		
1.	srcaddr-key	Edit	Remove
2.	dstaddr-key	Edit	Remove
3.	src-mask-key	Edit	Remove
4.	dst-mask-key	Edit	Remove
5.	src-subnet-key	Edit	Remove
6.	dst-subnet-key	Edit	Remove
7.	masked-srcaddr-key	Edit	Remove
8.	masked-dstaddr-key	Edit	Remove
9.	srcport-key	Edit	Remove
10.	dstport-key	Edit	Remove
Rows per page: <input type="text" value="10"/> Go to page: <input type="text" value="1"/> of 3 Pages Go 			

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:

- [BGP Attribute, page 2-13](#)
- [Bit Field, page 2-14](#)
- [Boolean, page 2-14](#)
- [Byte Array, page 2-14](#)
- [Customer Name, page 2-15](#)
- [Egress PE, page 2-15](#)
- [Ingress CE, page 2-16](#)
- [Integer, page 2-16](#)
- [Integer Range Map, page 2-17](#)
- [Interface SNMP Name, page 2-17](#)
- [IP Address, page 2-17](#)
- [IP Address Range Map, page 2-18](#)
- [Mac Address, page 2-18](#)
- [Masked IP Address, page 2-18](#)
- [Multi-Field Map, page 2-19](#)
- [Option Data, page 2-20](#)
- [Site Name, page 2-20](#)
- [String, page 2-21](#)
- [Subnet Address, page 2-21](#)

BGP Attribute

A **BGP Attribute** key builder looks up a BGP attribute from the Cisco NetFlow Collector 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	One of the following radio buttons: <ul style="list-style-type: none">• Complete AS Path• Well Known Name—Select from ORIGIN, AS_PATH, NEXT_HOP, MULTI_EXIT_DESC, LOCAL_PREF, ATOMIC_AGGREGATOR, AGGREGATOR, COMMUNITY, ORIGINATOR_ID, or CLUSTER_LIST• Integer Type ID.
Source address key	ID of a key builder that returns the source address for a complete AS path look up, otherwise disabled.
Destination address key	ID of a key builder that returns the destination address for querying the attribute.
Post-aggregation	Determines whether look ups are performed for each flow or at the end of the aggregation period; this should always be selected, otherwise attributes are queried from the Cisco NetFlow Collector 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	<i>Decimal</i> or <i>hexadecimal</i> .
Allow null value	If <i>not</i> selected, an error is logged if a flow does not contain the indicated field.

Boolean

A **Boolean** key builder maps flow values to **true**, **false**, or **undefined**. The Boolean key builder 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 containing the value of interest.
Allow null value	If <i>not</i> selected, an error is logged if a flow does not contain the indicated field.

Byte Array

A **Byte Array** key builder outputs bytes from flow data in hexadecimal format. The Byte Array key builder has the following attributes.

Attribute	Description
Output name	Column name in output.
Field	ID of the field to obtain from a flow.
Offset	Starting byte offset from the beginning of the field in the flow. Set to zero if not specified.
Length	Number of bytes of interest, from the offset to the end of field data if not specified.
Allow null value	If <i>not</i> selected, an error is logged if a flow does not contain the indicated field.

Customer Name

The **Customer Name** key builder resolves the customer name from the input interface field. 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 <i>not</i> selected, an error is logged if a flow does not contain the indicated field.

The Customer Name key builder requires configuration in the **config/vpn.conf** file. You must include one row to correspond to each PE device VPN interface that export NetFlow packets to this NFC server. The rows in this file contains five fields, in the following order: exporting device (PE) IP address, interface name, name of the site to which this interface is connected, CE to which this interface is connected, and customer name. These fields should be separated by commas. See the following example:

```
172.20.98.250, FastEthernet0/1.401, vpn1-branchB, CERouter-3, Cisco
172.20.98.250, FastEthernet0/1.601, vpn2-branchB, CERouter-4, IBM
172.20.98.248, FastEthernet2/1, vpn2-branchA, CERouter-2, IBM
172.20.98.246, FastEthernet0/1, vpn1-branchA, CERouter-1, Cisco
```

The exporting device (PE) IP address and interface name fields are required. You can include empty strings for the remaining fields in each row if those fields do not need to be resolved. For example, if you do not need to specify a site name, the site name fields can be left empty.



Note

Each row must contain four commas. Empty fields must be separated with commas.

Egress PE

The **Egress PE** key builder resolves the egress PE from the BGP nexthop field. 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 <i>not</i> selected, an error is logged if a flow does not contain the indicated field.

This key builder requires configuration in the **config/peList.conf** file. This file should include the loopback addresses or hostnames of all PEs in the network. See the following sample:

```
# This file is for the PE-PE traffic summary only # It should contain a list of IDs
for all PE devices in the provider network # ID of PE device can be either host name
or IP address
192.168.200.2
192.168.200.3
192.168.200.4
```

Ingress CE

The **Ingress CE** key builder resolves the ingress CE from the input interface field. 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 <i>not</i> selected, an error is logged if a flow does not contain the indicated field.

This key builder requires configuration in the **config/peList.conf** file. You must include one row to correspond to each PE device VPN interface that export NetFlow packets to this NFC server. The rows in this file contains five fields, in the following order: exporting device (PE) IP address, interface name, name of the site to which this interface is connected, CE to which this interface is connected, and customer name. These fields should be separated by commas. See the following example:

```
172.20.98.250, FastEthernet0/1.401, vpn1-branchB, CERouter-3, Cisco
172.20.98.250, FastEthernet0/1.601, vpn2-branchB, CERouter-4, IBM
172.20.98.248, FastEthernet2/1, vpn2-branchA, CERouter-2, IBM
172.20.98.246, FastEthernet0/1, vpn1-branchA, CERouter-1, Cisco
```

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	<i>Decimal</i> or <i>hexadecimal</i> .
Allow null value	If <i>not</i> 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 <i>not</i> 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.

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 <i>not</i> selected, an error is logged if a flow does not contain the indicated field.

IP Address

An **IP 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	<i>Standard notation</i> , <i>hostname</i> (via a DNS look up), or <i>integer</i> . Note: The integer format is obsolete and should not be used. It is retained for backwards compatibility.
Allow null value	If <i>not</i> selected, an error is logged if a flow does not contain the indicated field.

IP Address Range Map

An **IP 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 look up from flows.
Allow null value	If set to <i>false</i> (default) and a flow does not contain field, an error is logged. If set to <i>true</i> , 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 IP 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.

Mac Address

The Mac Address key builder reads and outputs an MAC address. 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 look up from flows.
Allow null value	If set to <i>false</i> (default) and a flow does not contain field, an error is logged. If set to <i>true</i> , the output value is empty and no error is logged.

Masked IP Address

The **Masked IP Address** key builder is obsolete and should not be used. It will be removed in a subsequent release.

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. After 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 look up 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.

Site Name

The **Site Name** key builder resolves the customer site name from the input interface field. 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 <i>not</i> selected, an error is logged if a flow does not contain the indicated field

This key builder requires configuration in the **config/vpn.conf** file. You must include one row to correspond to each PE device VPN interface that export NetFlow packets to this NFC server. The rows in this file contains five fields, in the following order: exporting device (PE) IP address, interface name, name of the site to which this interface is connected, CE to which this interface is connected, and customer name. These fields should be separated by commas. See the following example:

```
172.20.98.250, FastEthernet0/1.401, vpn1-branchB, CERouter-3, Cisco
172.20.98.250, FastEthernet0/1.601, vpn2-branchB, CERouter-4, IBM
172.20.98.248, FastEthernet2/1, vpn2-branchA, CERouter-2, IBM
172.20.98.246, FastEthernet0/1, vpn1-branchA, CERouter-1, Cisco
```

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.
Regex filter	If specified, the regular expression is applied to the string in flow data. The first matching sequence becomes the value of the key. If the regex contains one or more capturing groups, the first match is returned.
Allow null value	If <i>not</i> selected, an error is logged if a flow does not contain the indicated field

Subnet Address

A **Subnet Address** key builder obtains an IP address and mask from a flow, applies the mask to the address, and outputs a network address in the format n.n.n.n/m. 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.
Allow null value	If <i>not</i> 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 a NetFlow Collector output file.

Clicking on the **Value Builders** folder of the navigation tree displays a table of all existing value builders, as shown in [Figure 2-11](#). Click on the appropriate link to modify or remove a value builder.

Figure 2-11 Value Builders

Value Builders			
Add Value Builder			
Showing 1-10 of 10 records			
	Value Builder		
1.	flow-count-value	Edit	Remove
2.	packet-count-value	Edit	Remove
3.	byte-count-value	Edit	Remove
4.	flow-rate-value	Edit	Remove
5.	packet-rate-value	Edit	Remove
6.	byte-rate-value	Edit	Remove
7.	start-time-value	Edit	Remove
8.	end-time-value	Edit	Remove
9.	active-time-value	Edit	Remove
10.	max-burst-rate-value	Edit	Remove
Rows per page: <input type="text" value="10"/> Go to page: 1 of 1 Pages Go 			

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 2-12](#)), 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 2-12 Adding a Value Builder

Value Builder	
ID: *	active-time-value
Type:	<input type="text" value="Active Time"/>
Output Name:	<input type="text" value="activetime"/>
Start Time Field: *	<input type="text" value="FIRST_SWITCHED"/>
End Time Field: *	<input type="text" value="LAST_SWITCHED"/>
<input type="button" value="Modify"/> <input type="button" value="Remove"/>	

See 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 .

Directional Sum

The **Directional Sum** value builder obtains an integer value from a field in a flow and adds it to a count if the flow direction agrees with what you specify with the Egress attribute. It has the following attributes.

Attribute	Description
Output Name	Column name in output.
Field	ID of the integer field to obtain from a flow.
Egress	Boolean attribute to indicate if flow direction is egress or not.

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 Flow Byte Rate

The **Max Flow Byte Rate** value builder determines the byte rate for each received flow and outputs the highest value found for all flows in an aggregation period. This builder was referred to as Max Burst Rate in previous releases. 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.
Allow null value	If not selected and the flow does not contain the specified field, an error is logged.

Sum with Sampling Estimation

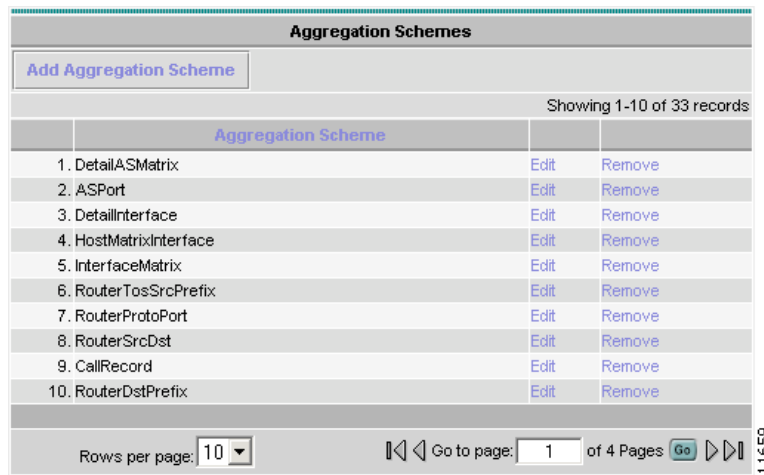
The **Sum with Sampling Estimation** value builder obtains an integer value from a field in a flow, multiplies by the sampling rate in effect, and adds the estimate to a count. If not used with V9 export, the value is not scaled because the sampling rate is not known. It has the following attributes.

Attribute	Description
Name	Column name in output.
Field	ID of the integer field to obtain from a flow.
Sampling Interval Builder ID	Always use the default value.
Allow null value	If not selected and the flow does not contain the specified field, an error is logged.

Aggregation Schemes

Aggregation schemes define the set of keys and values used for aggregation and that appear in the Cisco NetFlow Collector output files. Clicking on the **Aggregation Schemes** folder of the navigation tree displays a table of all existing aggregation schemes, as shown in [Figure 2-13](#). 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 2-13 Aggregation Schemes



Aggregation Schemes			
Add Aggregation Scheme			
Showing 1-10 of 33 records			
	Aggregation Scheme		
1.	DetailASMatrix	Edit	Remove
2.	ASPort	Edit	Remove
3.	DetailInterface	Edit	Remove
4.	HostMatrixInterface	Edit	Remove
5.	InterfaceMatrix	Edit	Remove
6.	RouterTosSrcPrefix	Edit	Remove
7.	RouterProtoPort	Edit	Remove
8.	RouterSrcDst	Edit	Remove
9.	CallRecord	Edit	Remove
10.	RouterDstPrefix	Edit	Remove
Rows per page: 10 Go to page: 1 of 4 Pages Go			

The **Add Aggregation Scheme** and **Modify Aggregation Scheme** in windows, as shown in [Figure 2-14](#), are identical with the exception that you cannot change the Aggregation Scheme ID on the Modify Aggregation Scheme window. Use this form to select key and value fields and click **Add** or **Modify** respectively to complete the operation. From the **Modify Aggregation Scheme** window you can also remove the currently displayed aggregation scheme.

Figure 2-14 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 2-15. 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 2-15 Filters

Filters			
Add Filter			
Showing 1-3 of 3 records			
	Filter	Edit	Remove
1.	filter2	Edit	Remove
2.	filter1	Edit	Remove
3.	filter3	Edit	Remove

Rows per page: 10 Go to page: 1 of 1 Pages Go

When adding and editing filters the windows 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 2-16](#). Click on the appropriate link to modify or remove a group. Click **Add Group** to bring up an empty form for defining a new source group.

Figure 2-16 NetFlow Export Source Groups

The **NDE Source Group** window, as shown in [Figure 2-17](#), 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** window you can also remove the currently displayed source group. See the “[Creating Source Groups](#)” section on page 4-24 for additional information about source groups.

Figure 2-17 NDE Source Group

NetFlow Export Source Access List

By default, Cisco NetFlow Collector collects from any device that sends NetFlow data to it. However, by specifying a NetFlow Export Source Access List, you can configure Cisco NetFlow Collector 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 2-18](#). 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. See the [“Creating Access Lists”](#) section on page 4-24 for additional information about configuring source access lists.

Figure 2-18 NDE Source Access List

NDE Source Access List

Action ☐ Permit ☒ Deny

Sources

[Add Source](#) [Remove Source](#)

Showing 0-0 of 0 records

Source
No records.

Rows per page: 10 Go to page: 1 of 1 Pages [Go](#)

Groups

Selection

Available Groups

Selected Groups

> Add >>

<< Remove <

[Modify](#)

12954

BGP Peer

Click the **BGP Peer** folder of the NFC UI navigation tree to display the configuration for the Cisco NetFlow Collector BGP peer, as shown in [Figure 2-19](#). Click on **Add Remote Peer** to specify a new BGP peer. If the BGP Identifier field is left blank, the BGP identifier of the Cisco NetFlow Collector 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. See the [“BGP Peer”](#) section on page 5-8 for additional information about BGP Peer configuration.

Figure 2-19 Local Peer Settings Window

Local Peer Settings	
BGP Port (1-65535):	<input type="text" value="179"/>
Command Port (1025-65535):	<input type="text" value="7777"/>
BGP Identifier:	<input type="text"/> (Leave blank for default value)
Session Timeout	<input type="text" value="60"/>
<input type="button" value="Submit"/>	

Remote Peers	
<input type="button" value="Add Remote Peer"/>	
Showing 0-0 of 0 records	
	Remote Peer Address ▼
No records.	
Rows per page:	<input type="text" value="10"/>
<input type="button" value="Go to page: 1 of 1 Pages"/> <input type="button" value="Go"/> <input type="button" value="Previous"/> <input type="button" value="Next"/>	

129569

Global

The settings in [Figure 2-20](#) affect how the Cisco NetFlow Collector 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 Cisco NetFlow Collector is restarted.

Figure 2-20 Global Parameters Window

Global Parameters	
Clean Up Interval:	<input type="text" value="24"/>
Clean Up Job:	<input type="text" value="\${NFC_DIR}/bin/nfc_clean"/>
Filesready File Directory:	<input type="text" value="\${NFC_DIR}/logs"/>
Output Field Delimiter:	<input type="text" value="vertical bar"/>
Start Output At Top Of The Hour:	<input checked="" type="checkbox"/>
Output Format:	<input type="text" value="mixed"/>
Mixed Output XML Header:	<input type="text" value="multi-line"/>
Include Date in Filenames:	<input type="checkbox"/>
Include GMT Offset in Filenames:	<input type="checkbox"/>
<input type="button" value="Submit"/>	

211333

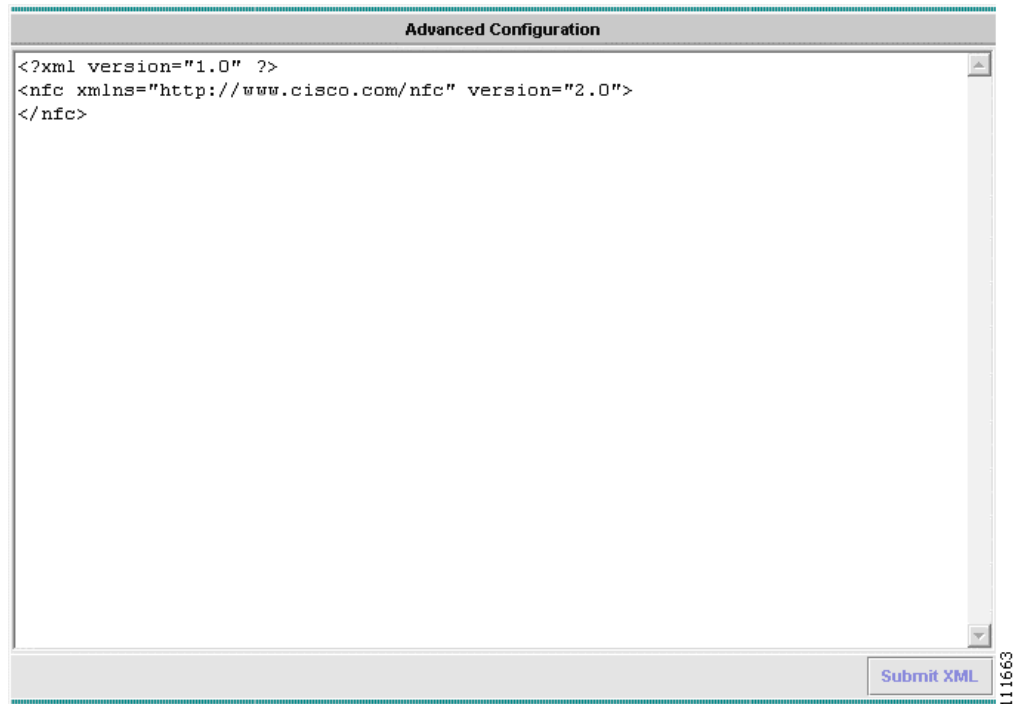
Advanced

The Advanced window lets you send any XML request to the collector. Clicking on the **Advanced** node in the NFC UI 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.

In limited cases where the configuration is more complex than the web-based UI supports, you will be directed to the **Advanced** window and the XML for the selected component 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.

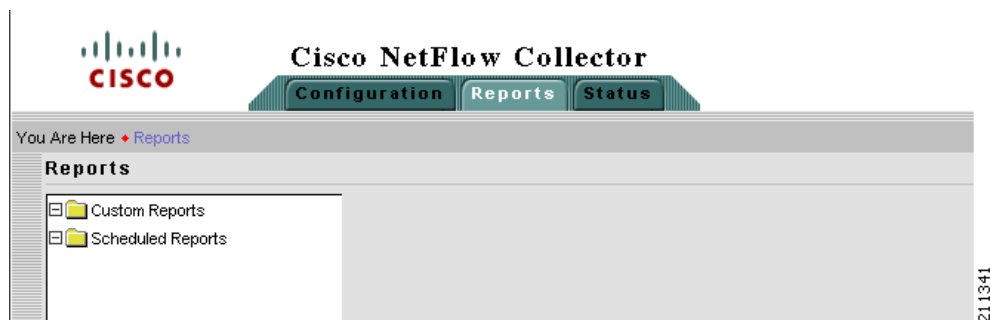
Figure 2-21 *Advanced Configuration Window*



Reports

Cisco NetFlow Collector reports are in effect a summary of the NetFlow Collector's aggregated output. NetFlow data is first aggregated into NetFlow Collector output files by the collector, and then the data in those files is further aggregated to generate a report. Reports are either custom (run immediately) or scheduled.

From the Cisco NetFlow Collector **Main** window, click the **Reports** tab. The Reports window appears, as shown in [Figure 2-22](#).

Figure 2-22 Reports Window

From this window you can select the following:

- [Custom Reports, page 2-32](#)
- [Scheduled Reports, page 2-37](#)

Custom Reports

Custom reports are generated on demand from the NetFlow Collector output files on the collector machine. From the **Custom Reports** window, as shown in [Figure 2-23](#), you can specify data that you want in the report and how you want it aggregated.

Figure 2-23 Custom Reports Window

Cisco NetFlow Collector

Configuration Reports Status

You Are Here: Reports > Custom Reports

Custom Reports

- Custom Reports
- Scheduled Reports

Custom Report

Start Date (dd MMM yyyy): 05 Apr 2007

Start Time (hh:mm:ss): 00:00:00

End Date (dd MMM yyyy): 05 Apr 2007

End Time (hh:mm:ss): 17:53:00

Relative Date and Time:

- ☒ Specified
- Hour: ☐ Current ☐ Previous ☐ Until now
- Day: ☐ Current ☐ Previous ☐ Until now
- Week: ☐ Current ☐ Previous ☐ Until now
- Month: ☐ Current ☐ Previous ☐ Until now

Devices:

- ☒ Combine devices
- ☐ Separate devices
- ☐ Single device:

Aggregator:

Keys:

Available Keys

Selected Keys

Values:

Available Values

Selected Values

Report Type: ☒ Top-N ☐ Bottom-N

N (maximum rows): 10

Ordered By:

Include Record "All": ☐ Yes ☒ No

Save as Template Generate XML Generate

The fields of the Custom Reports form are described in [Table 2-2](#).

211342

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, 01Jan2074 for January 1st, 2007.	The data for the report will come from Cisco NetFlow Collector 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 Cisco NetFlow Collector 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, 01Jan2007 for January 1st, 2007.	The data for the report will come from Cisco NetFlow Collector output files that were generated on or before this date.
End 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 Cisco NetFlow Collector output files that were generated at or before this time.
Relative Date and Time	Either the start and end date and time specified, or the calculated hour, day, week, or month relative to the current time. Also useful when creating report templates that are recalled and run later at the same relative time.	Selecting a relative time sets the start and end time relative to the current time. For example, if you select Current hour , the time range starts at the current hour of the day. If you select Previous hour , the last entire hour is shown. If you select Until now , the time range is set to end at the current time.

Table 2-2 Custom Reports Fields (continued)

Field	Value	Description
Devices	Combine devices , Separate devices , or Single device . For Single device the value should be the IP address of the device.	<p>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.</p> <p>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.</p> <p>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.</p> <p>In NFC Release 6.0, a selection box is provided for specifying a single device. You can select any device for which data is available. If the selections set is empty, no data is available for the selected aggregator.</p>
Aggregator	One of the defined aggregators	The report data will come from the Cisco NetFlow Collector 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.	<p>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.</p> <p>In NFC Release 6.0, three sets of value selections are provided. The first is the set of value columns available in output data. For integer values, the second and third sets allow per-minute and per-second rates calculated over the reporting period to be selected.</p>

Table 2-2 Custom Reports Fields (continued)

Field	Value	Description
Report Type	Top-N or Bottom-N	Specifies if the report shows the Top-N or Bottom-N values as determined by the Ordered By value selection.
N (Maximum Rows)	A positive integer, <i>N</i> , no greater than 2147483647. Default value is 10.	<p>The maximum number of rows the report should contain for each exporting device. The total number of unique records in all the Cisco NetFlow Collector data files being reported can be much greater than the number of the records one might want to present in a report. Use this field to limit the number of records contained in the report.</p> <p>You can sort all aggregated unique records in descending (or ascending) order, according to a user-specified value field, and present the first or last <i>N</i> records in the report. To show the relative magnitude of data that is not displayed, all records, not just those returned, can be optionally aggregated in to one record with key value of All.</p>
Include Record All	Yes or No . The default value is No .	Specifies whether to include the record with key value of All . If set to Yes , the All record will be calculated and appear in the report.

After filling in the fields in the **Custom Report** window, you can select one of the following actions:

- **Generate**—Runs the report in a separate browser window. A progress bar is shown until the report is displayed.
- **Generate XML**—Displays the underlying report XML in the browser window, which you can save as a file.
- **Save as Template**—Saves the report form contents as a template.

Report Templates

In NetFlow Collector Release 6.0, Report Templates replace and improve upon the Common Reports feature in previous releases. You can save the contents of a partially filled out custom report form as a template by selecting **Save as Template** and naming the template. You can then recall the template at any later time to run the report. This is particularly useful when used in conjunction with a relative date and time specification in the custom report form.

Report Templates are listed in the navigation tree under **Custom Reports**. When you select **Custom Reports** in the navigation tree, the list of Report Templates is displayed as displayed in [Figure 2-24](#). To run a report based on the template, select the template name in the navigation tree or select **Create report** in the report template list. You can remove a template by selecting **Remove** in the Report Template list.

Figure 2-24 Report Templates List

Custom Reports			
Create Report			
			Showing 1-2 of 2 records
	Report Template		
	1. Flows per minute	Create report	Remove
	2. SJ flows	Create report	Remove
Rows per page: 10		Go to page: 1 of 1 Pages Go	

If you select **Save as Template** in a custom report form that was created from a template, you can modify the template definition if you keep the existing template name when prompted for the name. You can also create a new template by specifying a new name.

For example, to create an hourly top-talkers report template for the previous hour, do the following:

-
- Step 1** Navigate **Reports > Custom Reports**.
 - Step 2** Click the **Previous hour** radio button to specify the **Relative Date and Time**.
 - Step 3** Select the **Devices** strategy to use. Specify either **Combined devices** or **Single device**.
 - Step 4** Select an aggregator whose aggregation scheme contains the **srcaddr** key and **octets** value.
 - Step 5** Select the **srcaddr** key and **octet** value.
 - Step 6** Click **Save as Template**.
 - Step 7** Enter the template name as **previous-hour-top-talkers** and click **OK**.

The template is saved. You can recall this template and run a report listing the previous hour's top talkers at any time.

Scheduled Reports

Scheduled reports are generated by the Report Generator on a regular basis. Beginning with Cisco NetFlow Collector 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 Window*

Scheduled Reports			
Add Scheduled Report			
			Showing 1-2 of 2 records
	Scheduled Report		
1.	DailyFlowStats	Edit	Remove
2.	HourlyHostMatrix	Edit	Remove
Rows per page: <input type="text" value="10"/> Go to page: <input type="text" value="1"/> of 1 Pages Go			

Clicking **Add Scheduled Report** brings up the **Add Scheduled Report** window to add a new scheduled report. Clicking **Edit** in any row in the list of scheduled reports displays the **Modify Scheduled Report** window to modify the selected scheduled report. Clicking **Remove** in any row deletes the selected schedule report. The **Add Scheduled Report** and **Modify Scheduled Report** windows, as shown in [Figure 2-26](#), are identical with the exception that you cannot change the **Report ID** on the **Modify Scheduled Report** window. 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

The screenshot shows the 'Add Scheduled Report' page in the Cisco NetFlow Collector UI. The breadcrumb trail is 'You Are Here > Reports > Add Scheduled Report'. The left sidebar shows a tree view with 'Custom Reports' and 'Scheduled Reports' (the latter is selected). The main content area is titled 'Add Scheduled Report' and contains the following fields:

- Scheduled Report ID:
- Report Frequency: ☒ Daily ☐ Hourly
- Start Time (hh:mm:ss):
- End Time (hh:mm:ss):
- Days To Keep:
- Devices: ☒ Combine devices, ☐ Separate devices, ☐ Single device:
- Aggregator:
- Keys:
 - Available Keys:
 - Selected Keys:
- Values:
 - Available Values:
 - Selected Values:
- Report Type: ☒ Top-N ☐ Bottom-N
- N (maximum rows):
- Ordered By:
- Include Record "All": ☒ Yes ☐ No
- Output Path:
- Submit button

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

- There is no Start Date, Start Time, End Date and End Time fields on Scheduled Report windows, 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 including 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.
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.	If Start Time and End Time are specified, the daily report will include data only for the time range within the day.
End 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.	If Start Time and End Time are specified, the daily report will include data only for the time range within the day.
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 .
Report Type	Top-N or Bottom-N	Specifies whether the report shows the Top-N or Bottom-N values as determined by the Ordered By value selection.

Table 2-3 *Scheduled Report Fields (continued)*

Field	Value	Description
N (maximum Rows)	A positive integer, <i>N</i> , no greater than 2147483647. Default value is 10.	The maximum number of rows the report should contain for each exporting device. The total number of unique records in all the NetFlow Collector data files being reported can be much greater than the number of the records you might 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 or last <i>N</i> records in the report. To show the relative magnitude of data that is not displayed, all records (not just those returned) can be optionally aggregated into one record with key value of All .
Ordered By	Value field name	The value field that determines report order. The first value field selected by default.
Include Record All	Yes or No. The default value is No.	Specifies whether to include the record with key value of All . If set to Yes , the All record will be calculated and appear in the report.

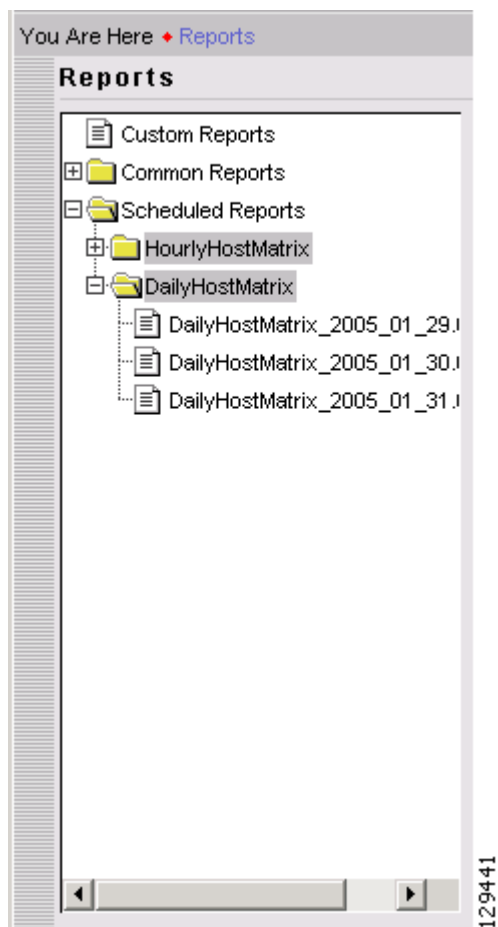
Displaying Scheduled Reports

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 Cisco NetFlow Collector 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

Cisco NetFlow Collector enables you to sort, graph, export, filter, and drill down on report data from the Report window, as shown in [Figure 2-22](#).

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 2-28](#). Click on the pie graph icon to generate a pie graph of that column's values, as shown in [Figure 2-29](#).

Figure 2-28 Sample Bar Graph

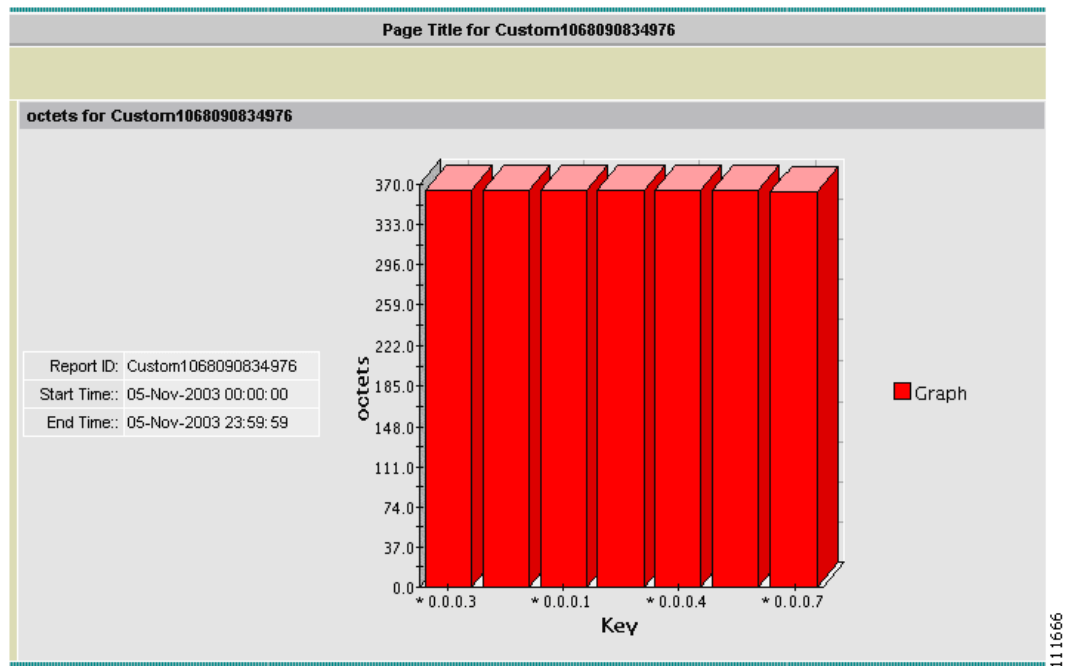
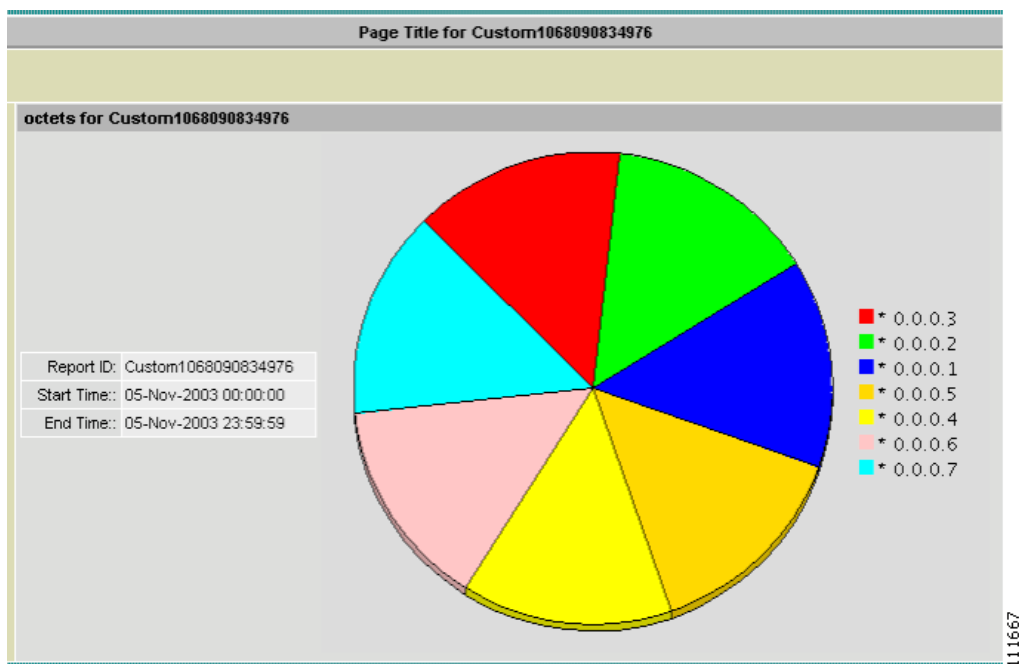


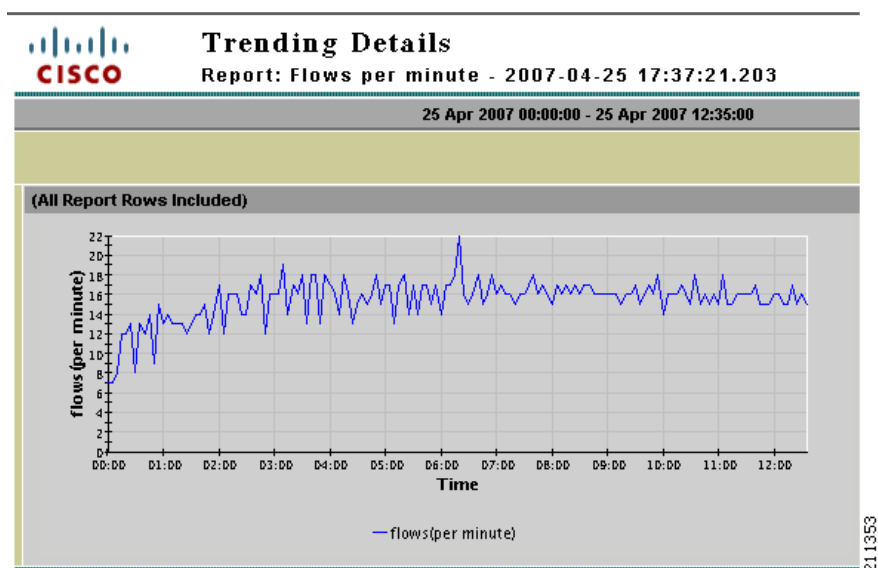
Figure 2-29 Sample Pie Graph



Trending

Trending reports can be launched from the Custom Report results window, as shown in [Figure 2-30](#). This allows you to see how one or more report values vary over time for the report period. To launch the Trending report, select a result row then select the **Trending** button.

Figure 2-30 Sample Trending Graph

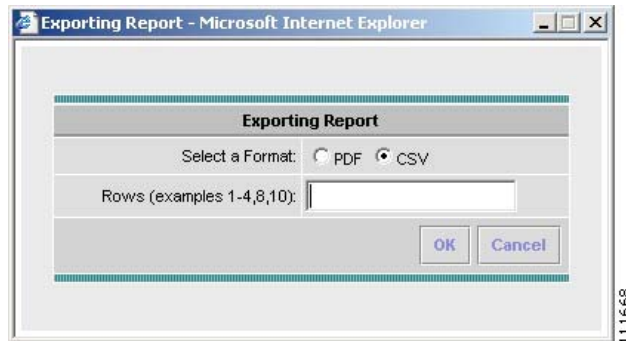


Export and Print

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

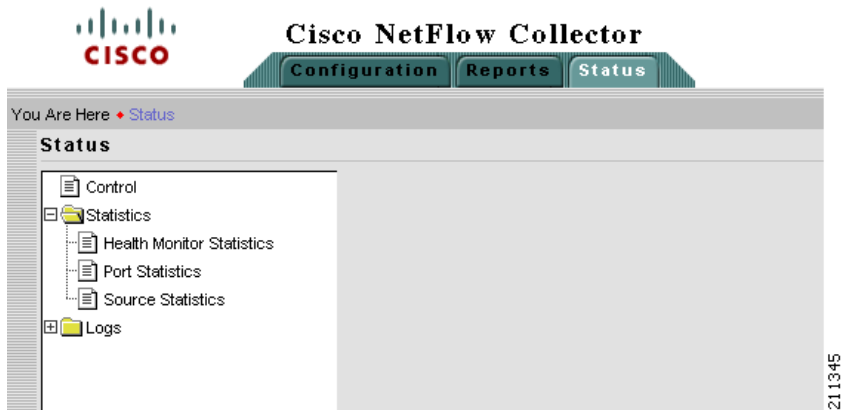
When the original Cisco NetFlow Collector output contains more keys than were used to generate a report, you can choose to *drill down* on the data by selecting a row, selecting an addition key, and clicking **Drill Down**. This will generate a new report where the original keys are fixed on the values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

Status

From the Status window you can view system health information about the collector. Such information includes running status, flows received statistics, flows missed statistics, and collector logs.

From the Cisco NetFlow Collector **Main** window, click the **Status** tab. The **Status** window appears, as shown in [Figure 2-32](#).

Figure 2-32 Status Window



From this window you can select the following:

- [Control](#), page 2-46
- [Statistics](#), page 2-46
- [Logs](#), page 2-49

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 can take affect. Most operations are not available when the collector is stopped.

Figure 2-33 Control Window



Statistics

The Cisco NetFlow Collector collects port and source statistics. The following sections describe Port Statistics and Source Statistics.

Health Monitor Statistics

Click on the **Health Monitor Statistics** folder of the **Statistics** navigation tree to display health and performance statistics for NetFlow Collector as shown in [Figure 2-34](#).

Figure 2-34 Health Monitor Statistics Window

Health Monitor Statistics			
Showing 1-6 of 6 records			
	Attribute Name	Current	Maximum
1.	CPU utilization (%)	1	33
2.	Disk utilization (%)	5	5
3.	Collector memory utilization (%)	1	6
4.	Packets processed (per second)	0	0
5.	Flows aggregated in current period	0	0
6.	Aggregation records in memory	0	0
Rows per page: <input type="text" value="10"/> Go to page: <input type="text" value="1"/> of 1 Pages <input type="button" value="Go"/> <input type="button" value="Refresh"/>			

Clicking **Refresh** updates the statistics displayed in the window. Also, the form refreshes automatically every 30 seconds. The table contains the following fields; each statistic contains both the current and maximum value.

Field	Description
CPU Utilization	CPU utilization percentage reported by the operating system.
Disk Utilization	Disk utilization percentage reported by the operating system for /opt/CSCOnfc/Data .
Collector Memory Utilization	Memory utilization percentage of the collection process, relative to the limit configured in /opt/CSCOnfc/config/nfcmem .
Packets Processes (per second)	Number of NetFlow packets processed per second by the collection process.
Flows Aggregated in Current Period	Number of flows aggregated in the current period; includes duplicate flows.
Aggregation Records in Memory	Number of aggregation records in memory; excludes duplicate flows.

Port Statistics

Click on the **Port Statistics** folder of the Statistics navigation tree to display statistics for the ports on which the Cisco NetFlow Collector has received data. See [Figure 2-35](#).

Figure 2-35 Port Statistics Window

Port Statistics					
Showing 1-1 of 1 records					
	Port / protocol ▼	Packets	Received	Missed	Out of sequence
1.	9991/udp	50509	80068	0	0
Rows per page: <input type="text" value="10"/> Go to page: <input type="text" value="1"/> of 1 Pages <input type="button" value="Go"/> <input type="button" value="Refresh"/>					

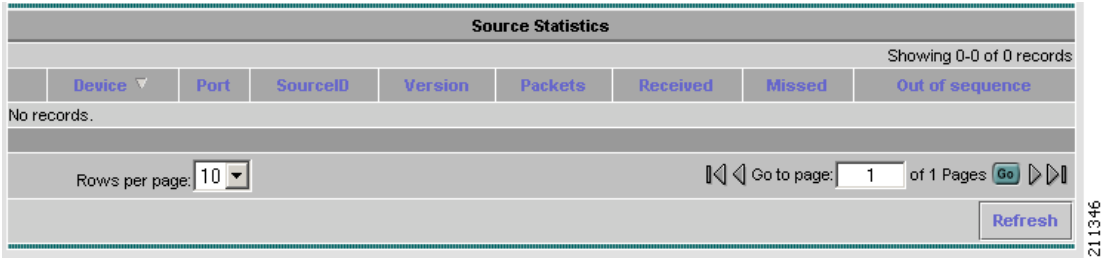
Clicking on **Refresh** updates the statistics shown. The table contains the following fields.

Field	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 Cisco NetFlow Collector has received data from. Source Statistics. See [Figure 2-36](#).

Figure 2-36 Source Statistics Window



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

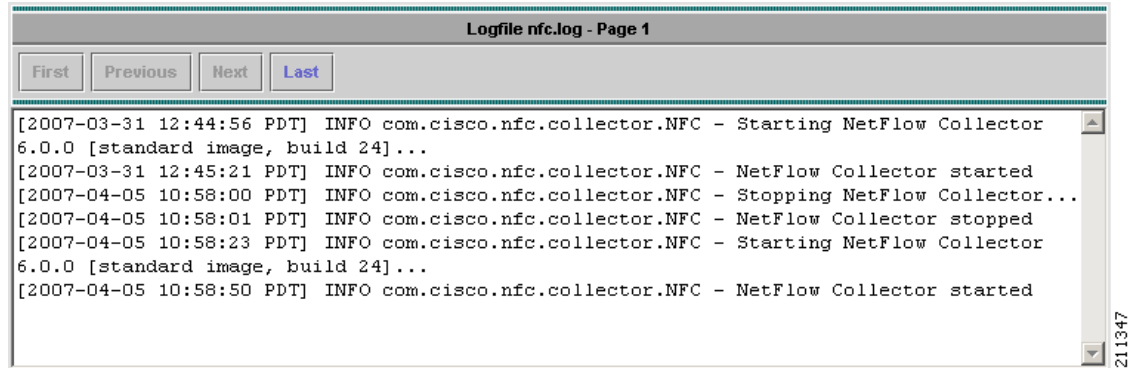
Field	Description
Device	IP address from where the data was received.
Port	Port and protocol
SourceID	source_id (V9) or engine_type and engine_id (other versions).
Version	Version of data received.
Packets	Number of packets received.
Received	Number of flows received.
Missed	Number of flows missed (estimate 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, Port, 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 2-37](#).

Figure 2-37 Viewing Logs in Web-based UI





CHAPTER 3

Understanding the NetFlow Collector Data File Format

This chapter tells you how to interpret the data collected and saved in the Cisco NetFlow Collector (NFC) data files.

This chapter includes the following sections:

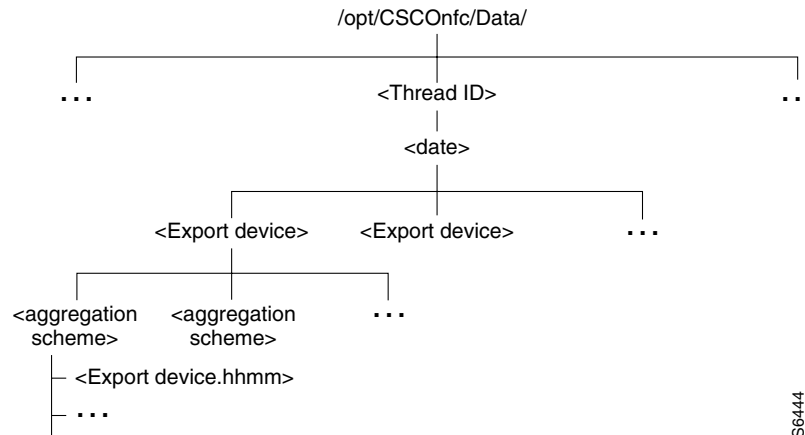
- [Data File Directory Structure, page 3-1](#)
- [Data Filenames, page 3-4](#)
- [Data File Format, page 3-4](#)
- [Options Data File Format, page 3-9](#)
- [Threshold Crossing Record File, page 3-10](#)
- [Using the filesready File to Track Data Files, page 3-10](#)

Data File Directory Structure

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

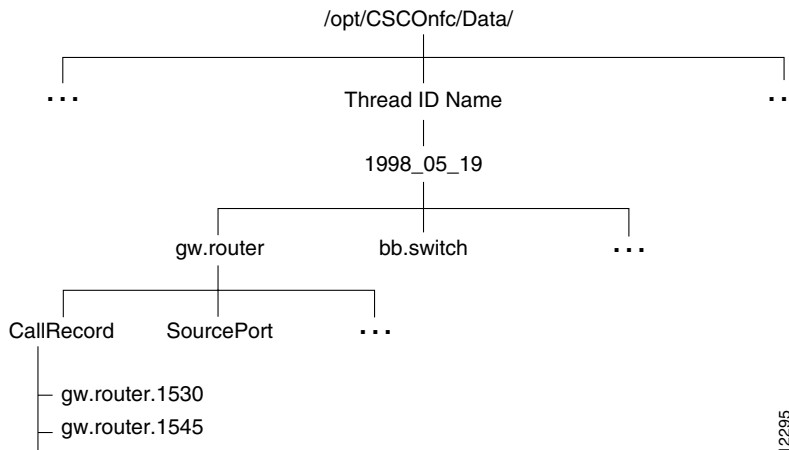
Default Data File Directory Structure

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, the NetFlow Collector 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

S6444

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

12295

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 NetFlow Collector Release 6, 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 NetFlow Collector 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 **nfc-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.

A new feature in NetFlow Collector Release 6.0 is the global property **filename-includes-gmt-offset** that can be configured in the **nfc-config.xml** file through the web-based user interface (UI) so that the time offset from GMT is appended to filenames. The general format is **+/-hhmm**, for example -0500 for Eastern Standard Time in the US.



Note

Without the **filename-includes-gmt-offset** setting, when the local time is turned back across a transition from daylight saving time to standard time, output files can be overwritten during the transition time period.

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

Table 3-1 Data Filename Format Fields and Descriptions

Field	Description
<i>name</i>	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.
<i>hhmm</i>	Time when the file was created in hours and minutes.
<i>YYYY_MM_DD</i>	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. NetFlow Collector Release 6 supports the following data file formats:

- **CSV:** Comma or vertical bar-separated header and data records. This format is backwards-compatible with NetFlow Collector Release 3 and 4.
- **Mixed:** XML header with comma or vertical bar-separated data records. New in NFC Release 6.0 is the global property **xml-header-format** that can be configured with the value **single-line** in the **nfc-config.xml** file using the UI. By setting the **xml-header-format** value to **single-line**, the XML header is written on one line, making it easier for applications to separate header information from CSV data.

- **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**

NetFlow Collector Release 6 reporting capability only works with mixed format output files.

Note that starting in NetFlow Collector Release 6, 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
<i>SOURCE</i>	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.
<i>FORMAT</i>	Tracks the version of the data file. Because NetFlow Collector, Release 4.0 uses tag 2 , that is the value used here.
<i>AGGREGATION</i>	The name of the aggregation scheme used to create this data file.
<i>PERIOD</i>	The data collection period, specified in minutes. Under some circumstances, NetFlow Collector might generate a data file before the current data collection period expires. In such a case, NetFlow Collector 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.
<i>STARTTIME</i>	The time in Coordinated Universal Time (UTC) seconds when this data collection period began.
<i>ENDTIME</i>	The time in UTC seconds when this data collection period ended.
<i>FLOWS</i>	The total number of NetFlow export records that are aggregated in this data file.
<i>MISSED</i>	The number of flow records that NetFlow Collector 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.

Table 3-2 Data File Header Keywords and Descriptions (continued)

Keyword	Description
<i>RECORDS</i>	The count of the aggregation records present in this data file.
<i>SAMPLEMODE</i> (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 Cisco NetFlow Collector” for more information.
<i>SAMPLEINTERVAL</i> (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).

Table 3-3 Data File Fields (continued)

Field	Description
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 include *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, etc.
value	Name and type of the values in the aggregation scheme. Supported types: integer, ipaddress, string, etc.
delimiter	Output field delimiter.
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, NetFlow Collector generates a data file periodically as determined by the period specified in aggregator definition. See the [“Creating an Aggregator” section on page 4-18](#).

Under certain circumstances, NetFlow Collector 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 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, NetFlow Collector 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 NetFlow Collector 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. See the [“Global Settings” section on page 4-28](#) 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 preceded 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.

Threshold Crossing Record File

If one or more threshold specifications are configured on an aggregator, output files are written containing the records with crossed thresholds. Because of the potentially large number of records associated with a threshold, the records themselves are not written in the threshold log. Instead, the log contains a reference to the threshold crossing record file.

The file name, path, and format are similar to normal output data files with these differences:

- The base directory is **/opt/CSCOnfc/threshold-data**
- The threshold id replaces the aggregation scheme id in the pathname
- Only **mixed** and **xml-only** output is supported
- The XML header root element is **nfc-threshold-info** instead of **nfc-output**; the header contains a threshold element with id and severity attributes.

Using the filesready File to Track Data Files

The NetFlow Collector periodically appends the absolute path names of data files that it has generated to a list in a log file named **filesready**. NetFlow Collector 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. See the “[Event Service](#)” section on page 4-34 for additional details.

After it finishes writing a new data file, NetFlow Collector 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 NetFlow Collector periodically appends the names of new options data files to a file in the NetFlow Collector logs directory. The name of this file is **\$NFC_DIR/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 in the future contain a **FORMAT** specification. However, this will not appear in NetFlow Collector, Release 6.

Threshold Filesready File

As with aggregated data files, NetFlow Collector periodically appends the names of new threshold data files to a file in the logs directory. The name of this file is:

\$NFCDIR/logs/threshold-filesready.<YYYY_MM_DD>.



CHAPTER 4

Customizing the Cisco NetFlow Collector

This chapter describes how to customize the Cisco NetFlow Collector (NFC) operations.

This chapter contains the following sections:

- [Configuration and Resource Files, page 4-1](#)
- [Data Collection and Aggregation, page 4-2](#)
- [Memory Usage, page 4-29](#)
- [Disk Space, page 4-31](#)
- [Process Watcher, page 4-32](#)
- [Event Service, page 4-34](#)
- [Scheduled Reports, page 4-35](#)
- [BGP Peer, page 4-38](#)
- [Interface Name Support, page 4-40](#)
- [Packet Log, page 4-41](#)
- [“Site In/Out Traffic Summary” section on page 4-42](#)

Configuration and Resource Files

To customize how the NetFlow Collector operates, you make changes and additions to the NetFlow Collector Engine configuration and resource files located in the **/opt/CSCOnfc/config** directory.

You can use a text editor to change any of these files. You can also use the NetFlow Collector User Interface (UI) to update the NetFlow Collector configuration. For details on using the NetFlow Collector UI, see [Chapter 2, “Using the NetFlow Collector User Interface.”](#)

XML Configuration and Schema

A basic working knowledge of XML and XML Schema is needed for editing the Netflow Collector configuration. An overview of XML is beyond the scope of this document. However, there are a number of resources for this available 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 **/opt/CSCOnfc/config** work together to define configuration information for Netflow Collector:

- **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 you typically do not change. You can apply additions and changes to the **nfc-config.xml** file, which NFC merges with the predefined file at runtime. If a definition appears in both files, changes in the **nfc-config.xml** file take precedence.

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

The configuration files for Netflow Collector 4.0 and earlier are no longer supported in Netflow Collector 6.0.

Data Collection and Aggregation

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

This section includes the following:

- [Overview of NetFlow Collector Release 6.0 Configuration Information, page 4-3](#)
- [Fields, page 4-4](#)
- [Keys and Values, page 4-4](#)
- [Creating an Aggregation Scheme, page 4-17](#)
- [Creating an Aggregator, page 4-18](#)
- [Creating a Filter, page 4-21](#)
- [Creating a Map, page 4-22](#)
- [Creating a Multi-Field Map, page 4-23](#)
- [Creating an Option Data Map, page 4-24](#)
- [Creating Source Groups, page 4-26](#)
- [Creating Access Lists, page 4-26](#)
- [Creating a Threshold, page 4-26](#)
- [Global Settings, page 4-28](#)
- [Setting the Time Zone, page 4-29](#)

Overview of NetFlow Collector Release 6.0 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**.

Table 4-1 *NetFlow Collector Configuration Elements*

Element	Purpose
global	Settings that affect the operation of the collector as a whole.
field-info	The name, type ID, content type, and alternate name for each field. 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	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	Defines an interpreter for each fixed-flow-packet-type as well as for NDE V9.
key-builders	Key definitions referenced by aggregation schemes.
value-builders	Value definitions referenced by aggregation schemes.
option-data-map	Defines option data map entries that are cached for use by option data key builders.
filters	Filter definitions referenced by aggregators.
aggregation-schemes	Aggregation scheme definitions listing which keys and values are aggregated and output; referenced by aggregators.
aggregators	Aggregator definitions.
nde-source-groups	Maps device IP addresses and/or hostnames to a group name. Devices in a group are aggregate together as one logical device.
nde-source- access-list	Access list for NDE source devices, to prevent devices from sending data to the collector.
flow-readers	Flow readers, one per NDE protocol (udp and sctp). By default, only upd is enabled. Uncomment the sctp entry to enable sctp on platforms where sctp is supported.
option-data-listeners	Option data processors.
cns-xml-interface	Settings for the configuration interface for the web-based user interface.
event-service	Defines supported transports for NetFlow Collector events and related options. Syslog and SNMP traps are the formats supported in NetFlow Collector, Release 6.0.
memory-monitor	Limits and logs memory usage by aggregation processing. Normally should not be modified.

Table 4-1 *NetFlow Collector Configuration Elements (continued)*

Element	Purpose
packet-log	Packet log settings for debugging. Normally should not be enabled.
disk-usage-monitor	Configures file systems 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. Unless new V9 fields are introduced, you will not typically add or change field definitions. Fields are important to consider because they are the building blocks upon which the keys and values referenced by aggregation schemes are built.

A field contains no information about what NetFlow Collector 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 NetFlow Collector, 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 the file **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 NetFlow Collector. 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 NetFlow Collector 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 the file `/opt/CSCOnfc/config/nfc-config-predefined.xml`; the exact syntax for defining each type appears in the XML schema in the `/opt/CSCOnfc/config/nfc-types.xsd` file. The web-based User Interface makes it easier to create new builder instances. See the “[Key Builders](#)” section on page 2-11 and the “[Value Builders](#)” section on page 2-21 for additional information.

Key Builder Types

address-range-map-key

The **address-range-map-key** builder reads an IP address and performs a map look up 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 look up 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 <i>false</i> (default) and a flow does not contain field, an error is logged. If set to <i>true</i> , 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 look up 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 NetFlow Collector BGP Peer. The look up result is BGP-4 attribute as described in RFC-1771. If the look up fails, the result is an empty string. See the “BGP Peer” section on page 5-8 for information about configuring and starting the NetFlow Collector BGP Peer.

The **bgp-attr-post-agg-key** builder performs a post-aggregation look up once at the end of each aggregation period. The **bgp-attr-key** builder performs a look up as each flow arrives, which reduces look up 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 look up 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 look up 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 NetFlow Collector BGP Peer. The results is a complete AS path from source to destination that is constructed from two separate look ups. The accuracy of the result assumes a symmetrical route to and from the source and the peer. If the look up fails, the result is an empty string.

The **bgp-complete-as-path-post-agg-key** builder performs a post-aggregation look up once at the end of each aggregation period. The **bgp-complete-as-path-key** builder performs a look up for as each flow arrives, which reduces look up 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 look up from flows.
least-significant-bit	Offset from zero of the least significant bit to include.
num-bits	Number of bits to include.
format	Either <i>decimal</i> (default) or <i>hex</i> .
is-null-allowed	If set to <i>false</i> (default) and a flow does not contain either field, an error is logged. If set to <i>true</i> , the output value is empty and no error is logged.

boolean-key

The **boolean-key** builder reads an integer flow value and interprets it as a boolean value. It outputs the value as **true** (for a flow value of 1), **false** (for a flow value of 2), or **undefined** (for all other flow values).

Key Builder	Description
name	Column name in output.
field	ID of the field to look up from flows.
is-null-allowed	If set to <i>false</i> (default) and a flow does not contain either field, an error is logged. If set to <i>true</i> , the output value is empty and no error is logged.

byte-array-key

The **byte-array-key** builder reads arbitrary data and outputs the data as hexadecimal digits.

Key Builder	Description
name	Column name in output.
field	ID of the field to look up from flows.
offset	Optional. Starting byte offset from the beginning of the field in the flow, defaults to zero if not specified.
length	Optional. Number of bytes of interest; from offset to the end of field data if not specified.
is-null-allowed	If set to <i>false</i> (default) and a flow does not contain either field, an error is logged. If set to <i>true</i> , the output value is empty and no error is logged.

customer-name-key

The **customer-name-key** builder resolves the customer name from the input interface field.

Key Builder	Description
name	Column name in output.
field	ID of the field to obtain from a flow.
is-null-allow	If <i>not</i> selected, an error is logged if a flow does not contain the indicated field.

egress-pe-key

The **egress-pe-key** builder resolves the egress PE from the BGP nexthop field.

Key Builder	Description
name	Column name in output.
Field	ID of the field to obtain from a flow.
is-null-allow	If <i>not</i> selected, an error is logged if a flow does not contain the indicated field.

ingress-ce-key

The **ingress-ce-key** builder resolves the ingress CE from the input interface field.

Key Builder	Description
name	Column name in output.
field	ID of the field to obtain from a flow.
is-null-allow	If <i>not</i> selected, an error is logged if a flow does not contain the indicated field.

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 look up from flows.
format	One of <i>standard-notation</i> (default), <i>hostname</i> , or <i>integer</i> . The <i>integer</i> format option will be removed in a future release and is not valid for IPV6 addresses.
is-null-allowed	If set to <i>false</i> (default) and a flow does not contain field, an error is logged. If set to <i>true</i> , the output value is empty and no error is logged.

If *format* is specified as *hostname*, a DNS look up is performed at output time to determine the hostname associated with the address. If the look up 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 look up from flows.
format	Either <i>decimal</i> (default) or <i>hex</i> .
is-null-allowed	If set to <i>false</i> (default) and a flow does not contain either field, an error is logged. If set to <i>true</i> , 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 look up 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 look up 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 <i>false</i> (default) and a flow does not contain either field, an error is logged. If set to <i>true</i> , 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 look up performance.

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

mac-address-key

The **mac-address-key** builder reads and outputs a MAC address.

Key Builder	Description
name	Column name in output; defaults to the field ID if not specified.
field	ID of the interface field to look up from flows.
is-null-allowed	If set to <i>false</i> (default) and a flow does not contain either field, an error is logged. If set to <i>true</i> , 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.

The masked-inetaddress-key builder is deprecated and will be removed in a subsequent release.

Key Builder	Description
name	Column name in output.
field	ID of the address field to look up from flows.
mask	Integer mask value to apply.
format	One of <i>standard-notation</i> (default), <i>hostname</i> , or <i>integer</i> . The <i>integer</i> format option will be removed in a future release and is not valid for IPV6 addresses.
is-null-allowed	If set to <i>false</i> (default) and a flow does not contain field, an error is logged. If set to <i>true</i> , 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 NetFlow Collector 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 4-23](#) for additional information about multi-field maps.

option-data-key

The **option-data-key** builder specifies a map look up from the option data map, described in the [“Creating an Option Data Map” section on page 4-24](#), 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 look up. The result of the look up 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 look up.

site-name-key

The **site-name-key** builder resolves the customer site name from the input interface field.

Key Builder	Description
name	Column name in output.
field	ID of the field to obtain from a flow.
is-null-allowed	If <i>not</i> selected, an error is logged if a flow does not contain the indicated field

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 look up from flows.
is-null-allowed	If set to <i>false</i> (default) and a flow does not contain either field, an error is logged. If set to <i>true</i> , the output value is empty and no error is logged.

subnet-address-key

The **subnet-address-key** builder reads both an Ip address and mask value from a flow and applies the mask to the address. The subnet address is output in a.b.c.d/n format.

Key Builder	Description
name	Column name in output.
field	ID of the address field to look up from flows.
mask-field	ID of the mask field to look up from flows.
is-null-allowed	If set to <i>false</i> (default) and a flow does not contain either field, an error is logged. If set to <i>true</i> , 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.

Value 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 look up from flows.
end-time-field	ID of the end time field to look up from flows.

directional-sum-value

The **directional-sum-value** builder obtains an integer value from a field in a flow and adds it to a count if the flow direction agrees with what user specifies with the Egress attribute.

Value Builder	Description
Output Name	Column name in output.
Field	ID of the integer field to obtain from a flow.
Egress	Boolean attribute to indicate if flow direction is egress or not.

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.

Value 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 look up from flows.

flow-count-value

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

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

max-flow-byte-rate-value

The **max-flow-byte-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. This builder was called max-burst-rate-value in previous releases.

Value 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 look up from flows.
end-time-field	ID of the end time field to look up from flows.
octets-field	ID of the count field to look up 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.

Value Builder	Description
name	Column name in output; defaults to the quantity-value-builder ID appended with <i>per-second</i> or <i>per-minute</i> 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 <i>seconds</i> or <i>minutes</i> .

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.

Value 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 look up from flows.

sampling-estimate-sum-value

The **sampling-estimate-sum-value** builder reads an integer from a flow, scales it by a sampling factor dynamically determined from option data, adds it to a running total, and outputs the total.

Value Builder	Description
name	Column name in output; defaults to the field ID if not specified.
sampling-interval-key	ID of option data key builder used to obtain sampling information. Predefined configuration contains the builder sampler-id-to-interval-key for this purpose.
field	ID of the field to look up from flows.

sum-value

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

Value Builder	Description
name	Column name in output; defaults to the field ID if not specified.
field	ID of the field to look up 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 NetFlow Collector 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
subnet-address-key	subnetaddress
macaddress-key	macaddress
address-range- map-key	string
boolean-key	boolean
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
byte-array-key	bytearray
interface-name-key	string
integer-range-map-key	string

Table 4-2 Aggregation Scheme Key Builders (continued)

Key Builder	Data Type
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
customer-name-key	string
egress-pe-key	depends on format attribute
ingress-ce-key	string
site-name-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
directional-sum	integer
end-time-value	utc
flow-count-value	integer
max-flow-byte-rate-value	integer
rate-value	integer
sampling-estimate-sum-value	integer
start-time-value	utc
sum-value	integer

For each data row in NetFlow Collector 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*.

See 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 NetFlow Collector, Release 3.x and 4.0, the fixed aggregation schemes for V1-V8 NDE were implemented in code and could not be changed. In NetFlow Collector 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).

Beginning with NetFlow Collector, Release 5, all aggregation is configurable. XML definitions are provided for all of the fixed aggregation schemes from NetFlow Collector, 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 NetFlow Collector, Release 3.x and 4.0 are defined.

Creating an Aggregator

An aggregator tells NetFlow Collector how to aggregate or 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.
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.
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 NetFlow Collector, 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 6 this is ascii-writer and optionally threshold writer. A writer contains additional elements as discussed below.

Table 4-5 *Aggregator Elements (continued)*

Element	Description
use-shortcut-address-as-source-ip	Optional, defaults to false. V7 and certain V8 NDE contain the field <i>router_sc</i> 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 <i>router_sc</i> 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 NetFlow Collector output file header.

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

Table 4-6 *ascii-writer Attributes*


Attribute	Description
output-base-dir	Base directory where output files are written, such as /opt/CSCOnfc/Data .
output-format	Optional; one of mixed , csv-only , or xml-only . If this is not specified, the global output-format setting determines the format. Note that reporting requires mixed output format.

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

Table 4-7 *ascii-writer Elements*

Element	Description
use-compression	Optional, defaults to false. If true, output files are compressed in <i>gzip</i> 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.

Table 4-7 *ascii-writer Elements (continued)*

Element	Description
filename-includes-gmt-offset	<p>Optional. Defaults to false for backwards compatibility. If true, the output file suffix includes the hours offset from GMT, for example -0500.</p> <p> Note Unless this attribute is set to true, a one-hour time window can exist during the transition from DST to standard time where the previous hour's files are overwritten one at a time.</p>
output-postprocessors	<p>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.</p> <p>The optional attribute <i>ignore-exit-status</i> 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.</p> <p>Output that the postprocessor writes to standard output and error is logged in /opt/CSCOnfc/logs/nfc.log.</p>

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

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, NetFlow Collector 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 *or* 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

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 of the address itself.

Creating a Multi-Field Map

The multi-field map introduced in NetFlow Collector, 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 NetFlow Collector, 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>
      </case>

      <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 look up 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, NetFlow Collector 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** builder; and the interface name string value returned by the **if-name-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** 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 NetFlow Collector output is an empty string, that is the column value in output is empty. Applications that process NetFlow Collector output files should expect this, because right after NetFlow Collector starts, there is always a window of time so that data flows arrive prior to option data flows.

**Note**

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 4-12](#).

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.

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. 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.

Creating a Threshold

A threshold creates a log and an optional event if one or more records in an NFC output file contain a value that crosses a threshold target value. One or more thresholds can be configured on an aggregator. NFC evaluates thresholds at the end of each aggregation period and creates an output file containing only those records that violated the threshold. A collector log in **nfc.log** and an optional event indicate that the violation occurred. See the [“Editing an Aggregator” section on page 2-8](#) for threshold file format details.

A threshold is specific to an aggregator. A threshold definition consists of a severity (**info** or **warn**) and an expression that evaluates to a boolean result. If the threshold expression evaluates to true for an output record, the threshold is violated. You can define a threshold to include logical operators and nested expressions of arbitrary complexity.

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

A condition returns a boolean result by comparing the value returned by a key or value builder against one or more values or ranges of values specified in the condition definition. The following condition operators are supported: **equals**, **not-equals**, **greater-than**, and **less-than**.

A threshold is configured on an aggregator by wrapping the threshold definition in a threshold-writer element, and adding the writer to the aggregator's list of writers.

In the following example a threshold violation with severity **warn** is logged if at any point in an aggregation period the source address key is 64.102.86.75 and the byte count value is greater than 300000.

```
<writers>
  <threshold-writer
    output-base-dir="${NFC_DIR}/threshold-data" output-format="mixed">
    <threshold id="byte-count-threshold" severity="warn">
      <expression op="and">
        <ipaddress-condition
          builder-id="srcaddr-key"
          builder-type="key" op="equals">
            <value>64.102.86.75</value>
          </ipaddress-condition>
        <integer-condition
          builder-id="byte-count-value"
          builder-type="value" op="greater-than">
            <value>300000</value>
          </integer-condition>
        </expression>
      </threshold>
    </threshold-writer>
    <!-- ... -->
  </writers>
```

Note that for efficiency, NFC performs *lazy* evaluations of threshold expressions. To illustrate, in the example above the source address condition is placed first within the **and** expression so that the byte count condition is evaluated only when the earlier source address condition is true.

Besides **and** and **or**, the additional expression operators **not-and** and **not-or** are provided which negate the evaluation result of an expression.

Multiple values and ranges can be specified within one condition if needed. 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.

Global Settings

Table 4-8 displays the global settings found in **nfc-config-predefined.xml**. In all pathnames in Global Settings, the top-level NetFlow Collector directory can be indicated as **\$/opt/CSCOnfc**. 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 <i>cleanup-interval</i> .
filesready-file-dir	Absolute path of directory where filesready files are written.
filename-includes-gmt-offset	If <i>true</i> , the output file name contains the offset from GMT. For example, -0500. If <i>false</i> or omitted, the GMT offset is not included. The default is <i>false</i> .
filename-includes-date	If <i>true</i> , the output file name contains the date in addition to the time of creation. If <i>false</i> or omitted, only the time is included. The default is <i>false</i> .
xml-header-format	If omitted or set to <i>multi-line</i> , the XML header in the output files is written on multiple lines. If set to <i>single-line</i> , the XML header is written on one line, making it easier for scripts to parse out the header.
output-field-delimiter	Separator character for fields in output, either a comma or vertical bar.
start-output-at-top-of-the-hour	If <i>true</i> , the time that an output file is first written after the collector restarts is calculated from the top of the hour. If <i>false</i> , output periods are relative to the collector start time.
output-format	One of <i>csv-only</i> , <i>mixed</i> , or <i>xml-only</i> . If <i>csv-only</i> , the NetFlow Collector output file header is compatible with NetFlow Collector, Release 3.x/4.0. If <i>mixed</i> , 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 <i>xml-only</i> , the entire output file is written as XML. Note that for output files to be compatible with the NetFlow Collector reporting feature, <i>output-format</i> must be <i>mixed</i> .

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.

To change the time zone used by NetFlow Collector from the local time zone, uncomment and update the TZ setting in `/opt/CSCOnfc/bin/nfcenv.sh`.

Memory Usage

NetFlow Collector memory requirements for collection and aggregation are determined by the following factors:

- Number of source devices
- Number and “uniqueness” of Netflow data flows 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, NetFlow Collector programs were compiled executables that in a poorly-scaled deployment would consume memory until all system memory was exhausted. However, because NetFlow Collector Release 6 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 NetFlow Collector 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 NetFlow Collector processes are consolidated in the file `$NFC_DIR/config/nfcmem`. For example, one of the largest potential consumers of memory is the collection/aggregation process. The default settings for the maximum amounts of memory this process can use is defined in `nfcmem` as follows:

```
COLLECTOR_MEM_MAX=-Xmx768M
```

-Xmx768M specifies that up to 768 megabytes can be allocated to aggregate Netflow data. This value was chosen as the initial default for a 2 gigabyte machine dedicated to running NetFlow Collector; other NetFlow Collector processes will consume additional memory.



Note

When increasing memory settings in `nfcmem`, you must verify that there is at least as much swap space configured on the system as the largest configured setting, so that NFC processes have sufficient virtual memory to start child processes.

System physical memory (RAM) must be greater than the combined maximum sizes of NFC processes configured in `nfcmem` with some additional headroom for system processes. If NFC runs in swap very significant performance degradation can occur.

The maximum memory value that can be configured for a process depends on the operating system:

- For 32-bit Red Hat Enterprise Linux with the standard kernel, the limit is approximately `-Xmx2600M` (2.6GB).
- For the 32-bit Red Hat Enterprise Linux with the hugemem kernel, the limit is approximately `-Xmx3600M` (3.6GB).
- For Solaris with the 32-bit JRE specified at install time, the limit is approximately `-Xmx3600M` (3.6GB).

Generating reports might require more memory than specified by default:

- To increase memory for running Custom Reports, for example if memory errors occur when running a custom report or are logged in `nferd.log`, increase the value of `RD_MEM_MAX`.
- To increase memory for running scheduled reports, for example if memory errors are logged to `nfre.log`, increase the value of `RE_MEM_MAX`.
- To increase memory for running command-line reports, for example if a memory error resulted from running the report, increase the value of `GEN_REPORT_MAX`.

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_MAX	Collection and aggregation (collector)
RD_MEM_INI, RD_MEM_MAX	Report daemon (rd) for custom reports
RE_MEM_INI, RE_MEM_MAX	Report engine (re) for scheduled reports
WEB_MEM_INI, WEB_MEM_MAX	Web server (web)
BGP_MEM_INI, BGP_MEM_MAX	BGP peer
GEN_REPORT_MEM_MIN, GEN_REPORT_MEM_MAX	Command-line reporting tool

The `nfcmem` setting `UDP_READER_POOL_ENTRIES` and `SCTP_READER_POOL_ENTRIES` specify 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 these default value are not typically updated, for especially high flow rates these can be increased at the cost of additional memory usage. For low flow rates, reducing these values can conserve memory, but with added risk of packets being dropped.

Disk Space

Depending on the volume of flow data being exported from the export devices and the NetFlow Collector thread attribute settings you use, NetFlow Collector can consume large amounts of disk space in a short period. NetFlow Collector 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 NetFlow Collector.

Aggregation

Aggregation schemes are used to define how you want NetFlow Collector 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 NetFlow Collector. 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 NetFlow Collector.

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 4-18](#) and [“Global Settings” section on page 4-28](#). You can limit disk space by doing any of the following:

- Change the number of days of output data to keep by changing the **DAYS_TO_KEEP** parameter in the default cleanup-job script `/opt/CSCOnfc/bin/nfc_clean_up_job.sh`. The default number of days is 7.
- Apply **gzip** compression to data files by enabling the **compression** option on each aggregator.

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



Note

Applying the **max-disk-usage** option is not usually recommended. Specifying **max-disk-usage** for an aggregator instead of using the periodic cleanup capability can result in a significant reduction in performance.

- Specify a different program or script to run periodically in the **cleanup-job** global parameter. You can specify the time period using the **cleanup-interval** global parameter.

Monitoring Disk Usage

NetFlow Collector 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 than one file system can be monitored if needed.

To monitor disk usage, create a **disk-usage-monitor** element in XML configuration. For each file system to be monitored, include a **monitor** element.

For example, with the following configuration the file system 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>
```

Process Watcher

The Process Watcher is responsible for starting, stopping, and restarting NetFlow Collector processes. It monitors NetFlow Collector processes and attempts to restart a process, up to the configured number of restarts, if the process exits unexpectedly with a non-zero return status.

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

[Table 4-10](#) displays the **nfccollector** command line arguments.

Table 4-10 *nfccollector Command Line Arguments*

Argument	Description
start all	Starts all managed processes marked for autostart.
shutdown	Gracefully stops all managed processes including the Process Watcher.
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.

Table 4-10 *nfcollector Command Line Arguments (continued)*

Argument	Description
stop <managed process id>	Stops the managed process with the corresponding ID attribute.
list	Displays a brief list of managed process status.
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.
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 NetFlow Collector 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 4-11](#).

Table 4-11 *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

Table 4-11 Configuration Elements (continued)

Child Element	Description	Required
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
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 NetFlow Collector 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

NetFlow Collector can be configured to generate an event for situations that include when the collector is started and it is stopped, an output file is created, or a threshold is crossed. Events can be sent via one or more event transports. Two event transports are supported in NFC Release 6: Syslogs and SNMP traps.



Note

The MIB used by NFC is not the official NFC MIB. The official NFC MIB will be released at a later time and the MIB Objects might change.

Configuration

To configure a Syslog event transport, add the following to the `/opt/CSCOnfc/config/nfc-config.xml` file:

```
<event-service>
  </syslog-event-transport>
</event-service>
```

To configure an SNMP trap event transport, add the following to the `/opt/CSCOnfc/config/nfc-config.xml` file:

```
<event-service>
  <snmp-event-transport>
    <target
      host="ipaddress"
      community="community string"
```

```

    port="port"/>
  </snmp-event-transport>
</event-service>

```

You can configure multiple target elements. You can also configure both **syslog-event-transport** and **snmp-event transport** within the event-service element.

**Note**

For syslog event transport to function on Red Hat Enterprise Linux, *syslogd* must be configured to receive messages from the network. Man pages for *syslogd* contain additional information.

Scheduled Reports

Report Generator is a separate process that produces reports based on NetFlow Collector data files. The configuration of the Report Generator allows generation of hourly and daily reports by performing further aggregation of the records in NetFlow Collector 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/nfcrc.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 4-12](#):

Table 4-12 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 NetFlow Collector 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 <i><others></i> . 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>
    ...
  </exporting-device>
</report>
```

```

<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

The NetFlow Collector Release 6 includes a Border Gateway Protocol (BGP) peer for supplementing NetFlow Collector 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 NetFlow Collector 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 NetFlow Collector BGP peer does a longest match on the destination address against advertised prefixes for the BGP peer.

Starting and Stopping the BGP Peer

If you want to have the BGP peer automatically started and stopped with the system, add the following line at the bottom of the `/etc/inittab` file .

On the Solaris platform:

```
bgp:3:respawn:/opt/CSCOnfc/bin/startbgp.sh
```

On the Linux platform:

```
bgp:35:respawn:/opt/CSCOnfc/bin/startbgp.sh
```

The init process will start and restart the BGP peer when the system enters the run level that NFC runs in.

Configuration

You can configure the BGP settings of the NetFlow Collector BGP peer in `NFC_DIR/config/nfcbgp.xml` with the elements shown in [Table 4-13](#). These XML elements are children of a `<bgp-peer>` root element.

Table 4-13 BGP Configuration Elements

Element	Description	Required
port	Port on which the NetFlow Collector BGP peer listens for BGP messages. Default: 179 (standard BGP port value).	No
command-port	Port on which the NetFlow Collector BGP peer listens for queries for BGP attributes. Default: 7777.	No
bgp-id	BGP identifier for the NetFlow Collector BGP peer. Default: IP address of collector machine.	No
path-separator	Separator between paths in the result of a complete AS path look up.	No
holdtime	Maximum amount of time in seconds allowed between messages before session is timed out. The NetFlow Collector 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 4-14](#). The `<bgp-session>` elements are also children of the root `<bgp-peer>` element.

Table 4-14 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 NetFlow Collector into a NDE source group, use this element to identify the group name used by NetFlow Collector.	No
nde-source	Specify an <i>nde-source</i> 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, <i>nde-source</i> should not be specified here. Note: In Release 5.0.3, <i>nde-source</i> 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

NetFlow Collector includes the option to map interface indices to interface names and allow the interface name to be an aggregation key. NetFlow Collector 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 configuration of this feature is `/opt/CSCOnfc/config/nfcifname.xml`. You must configure the SNMP read-only community string of a NDE source device if it differs from the default value. The syntax of this configuration is as follows:

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

In NFC Release 6, the default community string value can be updated by changing **public** to some other value:

```
<comm.-string>public</comm-string>
```


In order to reduce the amount of SNMP queries and enhance the application performance, NetFlow Collector caches the SNMP query results in memory. You can 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>
```

The SNMP **ifName** or **ifAlias** can be retrieved as the mapped interface name, instead of the default **ifDescr**. In order to retrieve **ifName** or **ifAlias**, change the value in the **<mib-object-to-query>** element to **ifName** or **ifAlias** as follows:

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

An OID (starting with .) can also be specified. For example .1.3.6.1.2.1.31.1.1.1.18 to retrieve **ifAlias**.

After changes are made in the **nfcifname.xml** file, you must restart NetFlow Collector for these changes to take effect.

Packet Log

Decoded NetFlow packets can be logged for debugging purposes. The system performance and disk space required for this are extremely high, so packet logging should not normally be enabled. To enable the packet log, add the following to the **/opt/CSCOnfc/config/nfc-config.xml** file:

```
<packet-log
  enabled="true"
  include-packet-hex-dump="false"
  base-dir="${NFC_DIR}/logs/packetlog"
>
  <port protocol="udp">9991</port>
  <port protocol="udp">9992</port>
  <device>64.102.86.75</device>
  <device>64.102.86.76</device>
</packet-log>
```

If the packet log is enabled, the file **/opt/CSCOnfc/logs/packetlog/yyyy-MM-dd-HH:mm:ss** is created each time NetFlow Collector is started. As packets arrive, they are decoded and logged to the file.

The port and device elements are optional and provide a way to limit which ports and devices for which NetFlow packets are logged.

Site In/Out Traffic Summary

Site In/Out Traffic Summary provides a summary of the total traffic going in and out of customer VPN sites connected to an MPLS core network via a Provider Edge Device (PE).

Site In/Out Traffic Summary requires the following be enabled:

- Ingress NetFlow
- Egress NetFlow
- MPLS egress NetFlow on the PE

Use the following commands:

ip flow ingress

ip flow egress

mpls netflow egress

On the NFC server, define an aggregator that uses the Site In/Out Traffic aggregation scheme. See the [“Aggregators” section on page 2-7](#) for details on how to define a new aggregator. Note that the aggregation scheme contains the Site Name key builder. See the [“Site Name” section on page 2-20](#) for the required configuration information.

You can generate a Site In/Out Traffic Summary report based on the results of this aggregator using the NFC custom report feature. See the [“Reports” section on page 2-31](#).



APPENDIX **A**

Troubleshooting the Cisco NetFlow Collector

This appendix provides helpful information and procedures in case you encounter problems while using the Cisco NetFlow Collector (NFC).

This appendix includes the following:

- [“Using the `nfc collector list` Command” section on page A-1](#)
- [“Using the `show-tech` Command to Capture Troubleshooting Information” section on page A-2](#)
- [“NetFlow Collector Tools and Utilities” section on page A-2](#)
- [“Solving NetFlow Collector Problems” section on page A-4](#)

Using the `nfc collector list` Command

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

`$NFC_DIR/bin/nfc collector list`

When invoked, the **`nfc collector list`** command displays status information about the Cisco NetFlow Collector, as in the following example:

```
rmiregistry: Running (pid: 13415)
nfcxml: Running (pid: 13403)
snmpd: Running (pid: 13425)
collection: Running (pid: 13405)
rd: Running (pid: 13404)
re: Not Running; autostart not configured
web: Running (pid: 7590)
```



Note

If the **`nfc collector list`** command lists that a process is not running but autostart is configured for that process, there may be a problem with the NetFlow Collector. See the [“Starting the Cisco NetFlow Collector User Interface” section on page 2-1](#) for information on how to start NetFlow Collector processes. Keep in mind that the **`re`** process for running scheduled reports is not autostarted by the process watcher unless you update the default process watcher configuration in `/opt/CSCOnfc/config/nfcpw.xml`.

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 NetFlow Collector is running.

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

NetFlow Collector Tools and Utilities

The utilities described in this section are typically used to troubleshoot NetFlow Collector 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](#)
- [get_bgp_rib Utility](#)
- [fdget Utility](#)
- [fdplayback 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 <i>UDP-port</i>	UDP port number on which flows are to be received. The default is 9991.
-c <i>count</i>	Number of flows to sample before calculating the incoming rate. The default is 100.
-s <i>socket-buffer</i>	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.

get_bgp_rib Utility

The **get_bgp_rib** utility displays the contents of the NetFlow Collector 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 is made obsolete by the **ndeget** 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 <i>UDP-port</i>	UDP port number on which flows are to be received. The default is 9991.
-s <i>socket-buffer</i>	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 NetFlow Collector 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.

Solving NetFlow Collector Problems

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

Symptom NetFlow Collector 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.



Note

In most cases it is recommended that you do not use the **max-disk-usage-megabytes** aggregator attribute value because calculating disk usage for a specific aggregator can consume significant system resources.

Symptom The export device is exporting NetFlow data to a port, but NetFlow Collector 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 NetFlow Collector. Use an unreserved port number in the range 1024 to 65535 (for example, 9995 or 9996) to export data to NetFlow Collector.

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

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

Recommended Action Running as the owner of NetFlow Collector.

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.

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. Run user scripts in the background by specifying a **front-end** script that simply launches the user script as a background process and exits immediately.

Symptom Can not generate a report using a specific aggregator.

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

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

Symptom After enabling SCTP, you are not able to log in to the NFC server via the web.

Possible Cause Enabling SCTP on a platform that does not support SCTP. To use SCTP, you must be running NFC on either the Red Hat Enterprise Linux release 4 (Update3) or Solaris 10 platforms.

Recommended Action Check the file **nfcscptrdr.log** for the reason you are not able to log in. If the problem is a result of enabling SCTP on an unsupported platform, you need to comment those lines back to the **opt/CSCOnfc/config/nfc-config.xml** file and restart the server. See the following example.

```
<!--
  <flow-readers>
    <default-flow-reader protocol="udp" />
    <default-flow-reader protocol="sctp" program="startsctpreader.sh" />
  </flow-readers>
-->
```

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

Possible Cause 1) Collector process is not running; 2) CNS/XML interface process 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 process and CNS/XML interface process 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 NetFlow Collector, 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 NetFlow Collector 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 NetFlow Collector, or update the access list entry for this device. See the [“Creating Access Lists” section on page 4-24](#) for more information.

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 can result if the file system containing `/opt/CSCOnfc` is full, or in case of a permission problem the wrong user started NetFlow Collector.

Recommended Action Check file system capacity (for example, `df -k /opt/CSCOnfc`). Verify that the user that started NetFlow Collector 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:

- Suspend processing incoming flows, memory used=xx%
- Discarded xx of yy total flows (zz%) this aggregation period due to memory constraints
- Output for aggregator *ID* has been queued multiple times without being written. Collector memory can 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 NetFlow Collector 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 Memory errors related to reporting in the `/opt/CSCOnfc/logs/nferd.log` file.

Possible Cause Report daemon maximum size is set too low for the time coverage and/or number of keys requested for the report.

Recommended Action Increase the report daemon maximum memory as outlined in the “[Memory Usage](#)” section on page 4-29 (the `RD_MEM_MAX` setting in `/opt/CSCOnfc/config/nfcmem`); decrease the time coverage and/or number of keys requested in the report.

Symptom Filter and threshold applets cannot be displayed in the browser. The screen area is blank.

Possible Cause The filter and threshold applets are compiled for the 1.5 Java Runtime Environment.

Recommended Action Download the latest 1.5 Java plugin from java.sun.com and configure it to run in the your browser.

Symptom Memory errors printed for command-line reporting.

Possible Cause Memory size for command-line reports is set too low for the time coverage and/or number of keys requested for the report.

Recommended Action Increase the command-line reports maximum memory as outlined in the “[Memory Usage](#)” section on page 4-29 (the `GEN_REPORT_MEM_MAX` setting in `/opt/CSCOnfc/config/nfcmem`); decrease the time coverage and/or number of keys requested in the report.

Symptom Collection process fails to start with the following error in `nfc.log`:

An exception occurred during XML configuration processing:
`com.cisco.nfc.collector.config.ConfigurationException: java.rmi.ConnectException:`
`Connection refused to host`

Possible Cause Insufficient system resources to start all NFC processes.

Recommended Action Verify that the NFC host is a dual-processor entry-level server with at least the minimum specifications as described in the *Cisco NetFlow Collector Installation and Configuration Guide*.



APPENDIX B

Logging

The Cisco NetFlow Collector, Release 6 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.

This appendix includes the following sections:

- [Configuration, page B-1](#)
- [Configuring the Logger from the Command Line, page B-2](#)
- [Rolling File Option for NFC Logs Files, page B-3](#)
- [Daily Rolling File Option for NFC Log Files, page B-3](#)

Configuration

All logging configurations come from files stored in the **NFC_DIR/config** directory. [Table B-1](#) displays the log configuration file for each component of NetFlow Collector, Release 6.

Table B-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
Report Daemon	NFC_DIR/config/nfcrd-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 NetFlow Collector 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 minimum logging level of NetFlow Collector 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.

Log configuration can be set so that log file size is limited or to enable rolling log files so that a new log file is created periodically. The log properties configuration script **/opt/CSCOnfc/bin/logconfig.sh** has been provided as a convenience for setting these options. The following options can be specified:

- **default** – reverts to default settings
- **filesize size[KB|MB] -maxfiles num** – limit the size and number of old files retained as specified
- **period hourly|twicedaily|daily|weekly|monthly** – enables rolling logs within the specified period.

For example:

```
$NFC_DIR/bin/logconfig.sh -filesize 200MB -maxfiles 3
```

Configuring the Logger from the Command Line

You can use the **logconfig.sh** script located in the **/opt/CSCOnfc/tools** directory to configure NFC log file settings. By default, the rollover file feature is disabled.



Note

The **nfcudprd.log** file 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|monthly
./logconfig.sh -period daily
```

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

```
./logconfig.sh -default
```



Note

The **logconfig.sh** script overwrites any changes that might 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 sections.

Rolling File Option for NFC Logs Files



Note

The NetFlow Collector GUI cannot display rollover log files. Only active log files are displayed in the web browser.

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. See [Table B-2](#).

Table B-2 **Rollover Schedule Elements**

Date Pattern	Rollover Schedule	Example
'.'yyyy-MM	Beginning of each month.	At midnight of May 31st, 2008 nfc.log will be copied to nfc.log.2008-05 . 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 2008, the file nfc.log will be copied to nfc.log.2008-23 . Logging for the 24th week of 2008 will be output to nfc.log until it is rolled over the next week.
".'yyyy-MMdd	Midnight each day.	At midnight, on March 8th, 2008, nfc.log will be copied to nfc.log.2008-03-08 . 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, 2008, nfc.log will be copied to nfc.log.2008-03-09-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, 2008, nfc.log will be copied to nfc.log.2008-03-09-10 . 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, 2008, nfc.log will be copied to nfc.log.2008-03-09-10-22 . Logging for the minute of 11:23 (9th of March) will be output to nfc.log until it is rolled over the next minute.



APPENDIX **C**

Cisco NetFlow Collector Sample Work Flow

This appendix contains a NetFlow Collector, Release 6.0 sample work flow to use as a reference.

- Step 1** Start the Cisco NetFlow Collector by entering:
/opt/CSCOnfc/nfcollector start all
- Verify in the **/opt/CSCOnfc/logs/nfc.log** file that the collection process started with no errors.
- Step 2** From a web browser window, open the URL:
http://<nfc-hostname>:8080/nfc
- Step 3** Log in as the web user you specified during installation.
- Step 4** Navigate to **Configuration > Aggregators**.
- Step 5** Click **Add Aggregator** and add an aggregator through the UI. Make sure there is NDE traffic arriving at the port to which the aggregator is listening either by checking the **nfc.log** file or the **Source Statistics** in the UI.
- Step 6** Wait until you see a new entry in the **filesready** file and view that data file. How long this takes depends on the period specified when the aggregator was created. 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|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 7 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 8 Go back to the web browser and navigate to **Reports > Custom Reports** and generate a report for the current hour on the data files produced by your aggregator.



APPENDIX **D**

NetFlow Fanout

The NetFlow Collector flow-fanout tool allows you to replicate NDE packets when multiple local or remote applications need to receive the raw NDE packets.



Note

The NetFlow Collector flow-fanout tool is only supported on the Red Hat Enterprise Linux (RHEL) versions 3.0 and 4.0 or Solaris 8, 9, or 10 platforms.

Command Line Syntax

The command line syntax for flow-fanout tool is:

```
startflowfanout.sh <listener> <destination1> <destination2> ...<destinationN>
```

For example:

```
startflowfanout.sh 10.0.0.1/10.0.0.2/9991 0/0/9992 10.0.0.1/10.0.0.5/9991
```

The above command will replicate the packets and send them to the two destinations defined.

The listener will listen for NDE packets on the interface assigned to IP address **10.0.0.1** on port **9991** from an exporting network device with IP address **10.0.0.2**.

The first destination **0/0/9992** is to a loopback address on port **9992**.

The second destination indicates that the packets will be sent to a remote host **10.0.0.5** on port **9991** with a source address of **10.0.0.1** which must be a valid local IP address.



Note

Use the **-p** option if you want to run the command without root user permission. You must run with root permission when using the spoof option.

To use the spoof feature, you need to add the **-s** option to the command line and run the command with root user. For example:

```
startflowfanout.sh -s 10.0.0.1/10.0.0.2/9991 0/10.0.0.5/9991
```

Configuration Note

If the flow-fanout tool is set to listen on a port on which a local aggregator is already configured, then no NDE packets will be sent to either the local or remote machine because the port is already in use by the local aggregator. In order for the flow-fanout tool to work correctly, you need to do the following:

- Set the fanout to a port different from any aggregator's port
- Configure the router to send the NDE packets to the fanout port.
- Set the local and remote NFC servers to the same time to avoid data discrepancies.



APPENDIX **E**

Cisco NetFlow Collector Binary Data File Format

The binary data file format that was backwards-compatible with NetFlow Collection Engine, Release 4.0 has been deprecated in NetFlow Collector Release 6. If you want to reduce the space consumed by output files, you should enable compression for ASCII output.



APPENDIX **F**

Cisco NetFlow Collector CNS/XML Interface

The CNS/XML interface in previous Cisco NetFlow Collector releases is deprecated in Release 6. The underlying transport mechanism has been replaced with Java RMI and the remote interfaces for this are not exposed.



INDEX

A

access list
 creating 4-26

Active Time value builder 2-23

active-time-value 4-13

address-range-map-key builder 4-5

Advanced configuration 2-30

aggregation scheme
 creating 4-17

Aggregation Schemes 2-25

Aggregation schemes 2-25

aggregator
 attributes 4-18
 creating 4-18
 elements 4-18

Aggregators 2-7

ascii-writer
 attributes 4-19
 elements 4-19

audience vii

B

Bar Graph 2-43

basic problems
 recommended actions A-4

BGP Attribute key builder 2-13

BGP Peer 2-29

bgp-attr-key builder 4-6

bgp-attr-post-agg-key builder 4-6

bgp-complete-as-path-key builder 4-6

binary data file format E-1

Bit Field key builder 2-14

bit-field-key builder 4-7

Boolean key builder 2-14

boolean-key builder 4-7

Border Gateway Protocol (BGP) peer 4-38

Byte Array key builder 2-14

byte-array-key builder 4-8

C

Cisco NetFlow Collector
 architecture 1-5
 Device and IOS Release Support 1-2
 functions 1-4, 1-5
 overview 1-1
 overview illustration 1-4

CNS NetFlow Collection Engine
 data file format 3-1

Collector subsystem (NFCollector) 1-6

command conventions viii

compatibility
 IOS software 1-2

config/peList.conf file
 configuring 2-15, 2-16, 2-20

configuration 2-5

configuration elements 4-3

Control 2-46

conventions, command viii

creating an aggregator 4-18

Custom Reports 2-32

Customer Name key builder 2-15

customer-name-key builder 4-8

customizing NetFlow Collector 4-1

D

daily rolling file option B-3

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 filenames 3-4

Directional Sum value builder 2-23

directional-sum-value builder 4-13

disk space

- data file options 4-31

disk space

- aggregation 4-31
- filters 4-31
- managing 4-31

disk usage

- monitoring 4-32

documentation

- obtaining ix

E

Egress PE key builder 2-15

egress-pe-key builder 4-8

End Time value builder 2-23

end-time-value 4-13

event service 4-34

export data 2-45

F

fdcount utility A-2

fdget utility A-3

fdplayback utility A-4

Fields 2-10

fields 4-4

filesready log file 3-10

filter

- creating 4-21

Filter report data 2-45

Filters 2-26

flow cache 1-2

Flow Count value builder 2-23

flow-count-value 4-14

flow-fanout tool D-1

flows

- defined 1-1

G

get_bgp_rib utility A-3

global settings 2-30

Global UI settings 2-30

H

Health Monitor Statistics 2-46

I

inetaddress-key builder 4-9

Ingress CE key builder 2-16

ingress-pe-key builder 4-8

Integer key builder 2-16

Integer Range Map key builder 2-17

interface name support 4-40

interface settings 2-3

Interface SNMP Name key builder 2-17

interface-name-key builder 4-10

interger-key builder 4-9

interger-range-map-key builder 4-10

IP address

- for configuration 1-4
- IP Address key builder 2-17
- IP Address Range Map key builder 2-18
- IP packets 1-1

J

- JRE 1.5 2-1

K

- key builder
 - address-range-map-key 4-5
 - BGP Attribute 2-13
 - bgp-attr-key 4-6
 - bgp-attr-post-agg-key 4-6
 - bgp-complete-as-path-key 4-6
 - bgp-complete-as-path-post-agg-key 4-6
 - Bit Field 2-14
 - bit-field-key 4-7
 - Boolean 2-14
 - boolean-key 4-7
 - Byte Array 2-14
 - byte-array-key 4-8
 - Customer Name 2-15
 - customer-name-key 4-8
 - Egress PE 2-15
 - egress-pe-key 4-8
 - inetaddress-key 4-9
 - Ingress CE 2-16
 - ingress-pe-key 4-8
 - Integer 2-16
 - Interface SNMP Name 2-17
 - interface-name-key 4-10
 - Interger Range Map 2-17
 - interger-key 4-9
 - interger-range-map-key 4-10
 - IP Address 2-17

- IP Address Range Map 2-18
- Mac Address 2-18
- mac-address-key 4-11
- Masked IP Address 2-18
- masked-inetaddress-key 4-11
- Multi-Field Map 2-19
- multi-field-map-key 4-11
- Option Data 2-20
- option-data-key 4-12
- Site Name 2-20
- site-name-key 4-12
- String 2-21
- string-key 4-12
- Subnet Address 2-21
- subnet-address-key 4-13
- key builder data types 4-15
- key builders 2-11
- keys and values 4-4

L

- log file
 - filesready 3-10
- log files
 - configuration from command line B-3
- logging B-1
- logging configurations B-1
- login screen 2-3
- logs 2-49

M

- Mac Address key builder 2-18
- mac-address-key builder 4-11
- map
 - creating 4-22
- Masked IP Address key builder 2-18
- masked-inetaddress-key 4-11

Max Flow Byte Rate value builder 2-24
 max-flow-byte-rate-value 4-14
 multi-field map
 creating 4-23
 Multi-Field Map key builder 2-19
 multi-field-map-key builder 4-11

N

ndget utility A-3
 NetFlow Collector
 Advanced configuration 2-30
 Aggregators 2-7
 architecture 1-5
 configuration 2-5
 configuration and resource files 4-1
 customizing 4-1
 Device and IOS Release Support 1-2
 Fields 2-10
 Filters 2-26
 functions 1-4, 1-5
 global settings 2-30
 key builders 2-11
 logging functions B-1
 overview 1-1
 overview illustration 1-4
 sample work flow C-1
 starting 2-1
 NetFlow data export
 hardware supported 1-2
 NetFlow Export Source Access List 2-28
 NetFlow Export Source Groups 2-27, 2-28
 NetFlow services
 device and IOS release support 1-2
 overview 1-1
 NFC 2-1, 2-3
 Advanced configuration 2-30
 Aggregators 2-7
 configuration 2-5

Fields 2-10
 Filters 2-26
 global settings 2-30
 key builders 2-11
 login screen 2-3
 reports 2-31
 NFC Collector 1-6
 nfcollector list command A-1

O

Option Data key builder 2-20
 Option Data Map
 creating 4-24
 option-data-key builder 4-12
 options data file
 format 3-9

P

packet log 4-41
 packets
 IP 1-1
 Pie Graph 2-44
 Port Statistics 2-47
 print report data 2-45
 Process Watcher 4-32

R

Rate value builder 2-24
 rate-value 4-14
 report data
 filter 2-45
 print 2-45
 Report Generator 4-35
 Report Templates 2-36
 reports 2-31

- Custom 2-32
- Scheduled 2-37
- rolling log files
 - ccreating B-3
- rollover schedule elements B-4

S

- sampling-estimate-sum-value builder 4-15
- Scheduled Reports 2-37
- Scheduled reports 2-37
- SCTP
 - port number configuration 1-4
- security guidelines ix
- show-tech command A-2
- Site In/Out Traffic Summary 4-42
- Site Name key builder 2-20
- site-name-key builder 4-12
- Sorting and Graphing 2-43
- source group
 - creating 4-26
- Source Statistics 2-48
- Start Time value builder 2-24
- starting 2-1
- starting NFC 2-1
- start-time-value 4-14
- Status 2-45
- String key builder 2-21
- string-key builder 4-12
- Subnet Address key builder 2-21
- subnet-address-key builder 4-13
- Sum value builder 2-24
- Sum with Sampling Estimation value builder 2-25
- sum-value builder 4-15
- support
 - obtaining ix

T

- threshold
 - creating 4-26
- thresholds 2-9
- time zone
 - setting 4-29
- traffic flows
 - description 1-1
- traffic statistics
 - information types 1-3
- Trending 2-44
- Trending Graph 2-44

U

- UDP
 - exporting NetFlow data to port 1-4
 - port number configuration 1-4
- user interface
 - Advanced 2-30
 - Aggregation schemes 2-25
 - Aggregators 2-7
 - BGP Peer 2-29
 - configuration 2-5
 - Fields 2-10
 - Filters 2-26
 - global settings 2-30
 - key builders 2-11
 - logs 2-49
 - navigation 2-4
 - NetFlow Export Source Access List 2-28
 - NetFlow Export Source Groups 2-27
 - Reports 2-31
 - statistics 2-46
 - Status 2-45
 - value builders 2-21

V

value builder

- Active Time 2-23
- active-time-value 4-13
- Directional Sum 2-23
- directional-sum-value 4-13
- End Time 2-23
- end-time-value 4-13
- Flow Count 2-23
- flow-count-value 4-14
- Max Flow Byte Rate 2-24
- max-flow-byte-rate-value 4-14
- Rate 2-24
- rate-value 4-14
- sampling-estimate-sum-value 4-15
- Start Time 2-24
- start-time-value 4-14
- Sum 2-24
- Sum with Sampling Estimation 2-25
- sum-value 4-15

value builder data types 4-16

value builders 2-21

Version 1 NetFlow export datagram

- description 1-3

Version 5 NetFlow export datagram

- description 1-3

Version 7 NetFlow export datagram

- description 1-3

Version 8 NetFlow export datagram

- description 1-3

Version 9 NetFlow export datagram

- description 1-3

X

XML

- header format 3-7

XML Schema 4-1