Network Manager IP Edition
Version 3 Release 9

Event Management Guide

R2E2

Network Manager IP Edition
Version 3 Release 9

Event Management Guide

R2E2

Note
Before using this information and the product it supports, read the information in"Notices" on page 247.

This edition applies to version 3, release 9, modification 0 of IBM Tivoli Network Manager IP Edition (5724-S45) and to all subsequent releases and modifications until otherwise indicated in new editions.

© Copyright IBM Corporation 2006, 2016.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

About this publication v
Intended audience v
What this publication contains v
Publications vii
Accessibility x
Tivoli technical training. x
Support information xi
Conventions used in this publication xi

Chapter 1. About polling the network . . 1
Poll policies 1
Poll policy parameters 2
Poll policy scope 2
Poll definitions 4
Poll definition parameters 4
Polling mechanisms 5
Poll definition types 8
Data labels 9
Ping polling properties and metrics 10
Multibyte data in poll definitions 10

Chapter 2. Enabling and disabling polls 11

Chapter 3. Creating polls 13
Creating fully featured poll policies 13
Creating simple poll policies. 19
Quick reference for poll policy creation based on custom data 20

Chapter 4. Creating new poll definitions 23
Creating basic threshold poll definitions 23
Creating generic threshold poll definitions 25
Creating chassis and interface ping poll definitions 27
Creating remote ping and link state poll definitions 29

Chapter 5. Changing polls 31
Changing poll policies. 31
Example poll policy 35
Changing poll definitions. 36
Changing basic threshold poll definitions . . . 36
Changing generic threshold poll definitions . . 38
Changing chassis and interface ping poll definitions. 40
Changing remote ping and link state poll definitions. 41
Example customized poll definition 43
Example basic threshold expression 44
Example generic threshold expression 44

Chapter 6. Deleting poll policies 47

Chapter 7. Deleting poll definitions . . 49

Chapter 8. Managing adaptive polling 51
Adaptive polling scenarios 51
Rapid confirmation that device is really down . . 51
Rapid confirmation of a threshold violation. . . 54
Creating adaptive polls 56

Chapter 9. Administering network polling 59
Administering polls 59
Speeding up ncp_poller startup by not checking SNMP credentials 59
Retrieving poll status 59
Enabling and disabling polls. 60
Refreshing polls 60
Copying polls across domains 61
Polling suspension options 61
Adjusting polling bandwidth 62
Configuring Link State polling 65
Configuring SNMP threshold polls 65
Administering multiple pollers 65
Multiple poller overview 66
Setting up additional pollers. 66
Removing a poller 69
Administering historical polled data 70
Storage capacity considerations 70
Increasing the storage limit for historical polled data 72
Deleting historical polled data 73
Monitoring poller capacity 74
Querying the status of entities 78

Chapter 10. Troubleshooting ping polling 81

Chapter 11. About event enrichment and correlation 83
Event enrichment 83
Quick reference for event enrichment. 83
Event filtering 85
Event states 93
Event handling 97
Example: Default enrichment of a Tivoli Netcool/OMNIbus trap event 115
Event Gateway plugins 118
Plugin descriptions 118
Plug-in subscriptions 128
Root-cause analysis 130
Quick reference for RCA 131
Precedence value 131
Poller entity 133

© Copyright IBM Corp. 2006, 2016

RCA and unmanaged status 134
RCA stitchers 136
Examples of root cause analysis 139
Checking topology paths used by RCA 148

Chapter 12. Configuring event enrichment 153
Configuring extra event enrichment 153
Modifications to the ObjectServer alerts.status table 153
Example: Enriching an event with main node device location 154
Example: Enriching an event with interface name 155
Configuring the ObjectServer update interval field 157
Using the OQL service provider to log into the Event Gateway databases 158
Querying the ObjectServer 158
Querying the NCIM database 158
Resynchronizing events with the ObjectServer . . 158
Configuring common Event Gateway properties 159

Chapter 13. Configuring Event Gateway plug-ins 161
Enabling and disabling plug-ins 161
Listing plug-in information 162
Modifying event map subscriptions 163
Setting plug-in configuration parameters 165
Configuring the SAE plug-in 166
Configuring summary field information in service-affected events 166
Adding SAE types to the SAE plug-in 167

Chapter 14. Configuring root-cause analysis 169
Configuring the poller entity 169
Configuring the maximum age difference for events 170
RCA considerations in a cross-domain network . . 170

Appendix A. Default poll policies . . . 173
Default ping policies 173
Default remote ping policies 173
Default SNMP threshold policies 174
Default SNMP link state policies 177
Poll policies used by reporting 177

Appendix B. Default poll definitions 179

Appendix C. Example trigger and clear thresholds 187

Appendix D. Syntax for poll definition expressions 189
eval statement syntax in threshold expressions . . 189
eval statement syntax for SNMP variables . . . 189
eval statement syntax for network entity variables 190
eval statement syntax for poll policy variables 191
eval statement syntax for poll definition variables 192
Operators in threshold expressions 192

Appendix E. Configuration of the Probe for Tivoli Netcool/OMNIbus . . 195
About the nco_p_ncpmonitor.props file 195
About the nco_p_ncpmonitor.rules file 196
nco_p_ncpmonitor.rules configuration reference 196
Example of rules file processing 197
Network Manager event data fields 199
alerts.status fields used by Network Manager 201

Appendix F. Network Manager event categories 207
Network Manager network events 208
Network Manager status events 208

Appendix G. Polling databases 213
NCMONITOR databases 213
SNMP tables for polling in the ncmonitor database 213
Ping polling status tables 216
OQL databases 225
config database for polling 225
profiling database for polling 228

Appendix H. Event enrichment databases 231
ncp_g_event database 231
The config database schema 231
ncp_g_event plug-in databases 236
RCA plug-in database 236
SAE plug-in database. 239
ncp_g_event plug-in database tables in ncmonitor 241

Appendix I. Network Manager glossary 243

Notices 247
Trademarks 249

Index 251

About this publication

IBM Tivoli Network Manager IP Edition provides detailed network discovery, device monitoring, topology visualization, and root cause analysis (RCA) capabilities. Network Manager can be extensively customized and configured to manage different networks. Network Manager also provides extensive reporting features, and integration with other IBM products, such as IBM Tivoli Application Dependency Discovery Manager, IBM Tivoli Business Service Manager and IBM Systems Director.

The *IBM Tivoli Network Manager IP Edition Event Management Guide* describes how to use IBM® Tivoli® Network Manager IP Edition to poll network devices.

Intended audience

This publication is intended for users, and system and network administrators who are responsible for configuring IBM Tivoli Network Manager IP Edition.

IBM Tivoli Network Manager IP Edition works in conjunction with IBM Tivoli Netcool/OMNIbus; this publication assumes that you understand how IBM Tivoli Netcool/OMNIbus works. For more information on IBM Tivoli Netcool/OMNIbus, see the publications described in "Publications" on page vii.

What this publication contains

This publication contains the following sections:

- Chapter 1, "About polling the network," on page 1

  Describes poll policies and poll definitions, and how they interact to create a network poll.
- Chapter 2, "Enabling and disabling polls," on page 11

  Describes how to enable and disable polls.
- Chapter 3, "Creating polls," on page 13

  Describes how to create polls, both by copying an existing poll and using the Poll Policy Wizard.
- Chapter 4, "Creating new poll definitions," on page 23

  Describes how to create new poll definitions.
- Chapter 5, "Changing polls," on page 31

  Describes how to change polls.
- Chapter 6, "Deleting poll policies," on page 47

  Describes how to delete poll policies when they are no longer required.
- Chapter 7, "Deleting poll definitions," on page 49

  Describes how to delete poll definitions when they are no longer required.
- Chapter 8, "Managing adaptive polling," on page 51

  Adaptive polls dynamically react to events on the network. The chapter describes adaptive polls that manage a wide range of network problem scenarios.
- Chapter 9, "Administering network polling," on page 59

© Copyright IBM Corp. 2006, 2016

Describes how to use the command-line interface to manage multiple pollers, copy network polls across network domains, and suspend network polling.

- Chapter 10, “Troubleshooting ping polling,” on page 81

  Describes how to ensure that the important IP addresses in your network are being polled as expected by Network Manager.
- Chapter 11, “About event enrichment and correlation,” on page 83

  Describes how the Event Gateway performs event enrichment, and how events are passed to plug-in processes such as root-cause analysis (RCA) and failover, which take further action based on the data in the enriched event. Also describes the mechanism by which the enriched event is passed back to the ObjectServer.
- Chapter 12, “Configuring event enrichment,” on page 153

  Describes how to configure the way an event is processed as it passes through the Event Gateway.
- Chapter 13, “Configuring Event Gateway plug-ins,” on page 161

  Describes how to configure the Event Gateway plug-ins.
- Chapter 14, “Configuring root-cause analysis,” on page 169

  Describes how to configure the Event Gateway RCA plug-in.
- Appendix A, “Default poll policies,” on page 173

  Describes the poll policies that are included with an installation of IBM Tivoli Network Manager IP Edition
- Appendix B, “Default poll definitions,” on page 179

  Describes the poll definitions that are included with an installation of IBM Tivoli Network Manager IP Edition
- Appendix C, “Example trigger and clear thresholds,” on page 187

  Provides example threshold formulas to set up the clear and trigger thresholds for generic threshold poll definitions.
- Appendix D, “Syntax for poll definition expressions,” on page 189

  Reference information to support building of complex threshold expressions to use in basic and generic threshold poll definitions.
- Appendix E, “Configuration of the Probe for Tivoli Netcool/OMNIbus,” on page 195

  Describes the Probe for Tivoli Netcool/OMNIbus, the probe that enables events generated by the Network Manager IP Edition polls to be sent to the Tivoli Netcool/OMNIbus ObjectServer.
- Appendix F, “Network Manager event categories,” on page 207

  The events that are raised by Network Manager fall into two categories: events about the network being monitored and events about Network Manager processes. This appendix provides more information on these events.
- Appendix G, “Polling databases,” on page 213

  Describes the structure of databases used for polling.
- Appendix H, “Event enrichment databases,” on page 231

  Describes the structure of databases used for event enrichment.

Publications

This section lists publications in the Network Manager library and related documents. The section also describes how to access Tivoli publications online and how to order Tivoli publications.

Your Network Manager library

The following documents are available in the Network Manager library:

- *IBM Tivoli Network Manager IP Edition Release Notes*, GI11-9354-00

  Gives important and late-breaking information about IBM Tivoli Network Manager IP Edition. This publication is for deployers and administrators, and should be read first.

- *IBM Tivoli Network Manager Getting Started Guide*, GI11-9353-00

  Describes how to set up IBM Tivoli Network Manager IP Edition after you have installed the product. This guide describes how to start the product, make sure it is running correctly, and discover the network. Getting a good network discovery is central to using Network Manager IP Edition successfully. This guide describes how to configure and monitor a first discovery, verify the results of the discovery, configure a production discovery, and how to keep the network topology up to date. Once you have an up-to-date network topology, this guide describes how to make the network topology available to Network Operators, and how to monitor the network. The essential tasks are covered in this short guide, with references to the more detailed, optional, or advanced tasks and reference material in the rest of the documentation set.

- *IBM Tivoli Network Manager IP Edition Product Overview*, GC27-2759-00

  Gives an overview of IBM Tivoli Network Manager IP Edition. It describes the product architecture, components and functionality. This publication is for anyone interested in IBM Tivoli Network Manager IP Edition.

- *IBM Tivoli Network Manager IP Edition Installation and Configuration Guide*, SC27-2760-00

  Describes how to install IBM Tivoli Network Manager IP Edition. It also describes necessary and optional post-installation configuration tasks. This publication is for administrators who need to install and set up IBM Tivoli Network Manager IP Edition.

- *IBM Tivoli Network Manager IP Edition Administration Guide*, SC27-2761-00

  Describes administration tasks for IBM Tivoli Network Manager IP Edition, such as how to administer processes, query databases and start and stop the product. This publication is for administrators who are responsible for the maintenance and availability of IBM Tivoli Network Manager IP Edition.

- *IBM Tivoli Network Manager IP Edition Discovery Guide*, SC27-2762-00

  Describes how to use IBM Tivoli Network Manager IP Edition to discover your network. This publication is for administrators who are responsible for configuring and running network discovery.

- *IBM Tivoli Network Manager IP Edition Event Management Guide*, SC27-2763-00

  Describes how to use IBM Tivoli Network Manager IP Edition to poll network devices, to configure the enrichment of events from network devices, and to manage plug-ins to the Tivoli Netcool/OMNIbus Event Gateway, including configuration of the RCA plug-in for root-cause analysis purposes. This publication is for administrators who are responsible for configuring and running network polling, event enrichment, root-cause analysis, and Event Gateway plug-ins.

- *IBM Tivoli Network Manager IP Edition Network Troubleshooting Guide*, GC27-2765-00

  Describes how to use IBM Tivoli Network Manager IP Edition to troubleshoot network problems identified by the product. This publication is for network operators who are responsible for identifying or resolving network problems.
- *IBM Tivoli Network Manager IP Edition Network Visualization Setup Guide*, SC27-2764-00

  Describes how to configure the IBM Tivoli Network Manager IP Edition network visualization tools to give your network operators a customized working environment. This publication is for product administrators or team leaders who are responsible for facilitating the work of network operators.
- *IBM Tivoli Network Manager IP Edition Management Database Reference*, SC27-2767-00

  Describes the schemas of the component databases in IBM Tivoli Network Manager IP Edition. This publication is for advanced users who need to query the component databases directly.
- *IBM Tivoli Network Manager IP Edition Topology Database Reference*, SC27-2766-00

  Describes the schemas of the database used for storing topology data in IBM Tivoli Network Manager IP Edition. This publication is for advanced users who need to query the topology database directly.
- *IBM Tivoli Network Manager IP Edition Language Reference*, SC27-2768-00

  Describes the system languages used by IBM Tivoli Network Manager IP Edition, such as the Stitcher language, and the Object Query Language. This publication is for advanced users who need to customize the operation of IBM Tivoli Network Manager IP Edition.
- *IBM Tivoli Network Manager IP Edition Perl API Guide*, SC27-2769-00

  Describes the Perl modules that allow developers to write custom applications that interact with the IBM Tivoli Network Manager IP Edition. Examples of custom applications that developers can write include Polling and Discovery Agents. This publication is for advanced Perl developers who need to write such custom applications.
- *IBM Tivoli Monitoring for Tivoli Network Manager IP User's Guide*, SC27-2770-00

  Provides information about installing and using IBM Tivoli Monitoring for IBM Tivoli Network Manager IP Edition. This publication is for system administrators who install and use IBM Tivoli Monitoring for IBM Tivoli Network Manager IP Edition to monitor and manage IBM Tivoli Network Manager IP Edition resources.

Prerequisite publications

To use the information in this publication effectively, you must have some prerequisite knowledge, which you can obtain from the following publications:

- *IBM Tivoli Netcool/OMNIbus Installation and Deployment Guide*, SC23-9680

  Includes installation and upgrade procedures for Tivoli Netcool/OMNIbus, and describes how to configure security and component communications. The publication also includes examples of Tivoli Netcool/OMNIbus architectures and describes how to implement them.
- *IBM Tivoli Netcool/OMNIbus User's Guide*, SC23-9683

  Provides an overview of the desktop tools and describes the operator tasks related to event management using these tools.
- *IBM Tivoli Netcool/OMNIbus Administration Guide*, SC23-9681

Describes how to perform administrative tasks using the Tivoli Netcool/OMNIbus Administrator GUI, command-line tools, and process control. The publication also contains descriptions and examples of ObjectServer SQL syntax and automations.

- *IBM Tivoli Netcool/OMNIbus Probe and Gateway Guide*, SC23-9684

  Contains introductory and reference information about probes and gateways, including probe rules file syntax and gateway commands.

- *IBM Tivoli Netcool/OMNIbus Web GUI Administration and User's Guide* SC23-9682

  Describes how to perform administrative and event visualization tasks using the Tivoli Netcool/OMNIbus Web GUI.

Accessing terminology online

The IBM Terminology Web site consolidates the terminology from IBM product libraries in one convenient location. You can access the Terminology Web site at the following Web address:

http://www.ibm.com/software/globalization/terminology

Accessing publications online

IBM posts publications for this and all other Tivoli products, as they become available and whenever they are updated, to the IBM Knowledge Center Web site at:

http://www-01.ibm.com/support/knowledgecenter/

Network Manager documentation is located under the **Cloud & Smarter Infrastructure** node on that Web site.

Note: If you print PDF documents on other than letter-sized paper, set the option in the **File** > **Print** window that allows your PDF reading application to print letter-sized pages on your local paper.

Ordering publications

You can order many Tivoli publications online at the following Web site:

http://www.elink.ibmlink.ibm.com/publications/servlet/pbi.wss

You can also order by telephone by calling one of these numbers:

- In the United States: 800-879-2755
- In Canada: 800-426-4968

In other countries, contact your software account representative to order Tivoli publications. To locate the telephone number of your local representative, perform the following steps:

1. Go to the following Web site:

   http://www.elink.ibmlink.ibm.com/publications/servlet/pbi.wss
2. Select your country from the list and click **Go**. The Welcome to the IBM Publications Center page is displayed for your country.
3. On the left side of the page, click **About this site** to see an information page that includes the telephone number of your local representative.

Accessibility

Accessibility features help users with a physical disability, such as restricted mobility or limited vision, to use software products successfully.

Accessibility features

The following list includes the major accessibility features in Network Manager:

- The console-based installer supports keyboard-only operation.
- The console-based installer supports screen reader use.
- Network Manager provides the following features suitable for low vision users:
  - All non-text content used in the GUI has associated alternative text.
  - Low-vision users can adjust the system display settings, including high contrast mode, and can control the font sizes using the browser settings.
  - Color is not used as the only visual means of conveying information, indicating an action, prompting a response, or distinguishing a visual element.
- Network Manager provides the following features suitable for photosensitive epileptic users:
  - Web pages do not contain anything that flashes more than two times in any one second period.

The accessibility of the Network Manager Knowledge Center is described in the Knowledge Center itself.

Extra steps to configure Internet Explorer for accessibility

If you are using Internet Explorer as your web browser, you might need to perform extra configuration steps to enable accessibility features.

To enable high contrast mode, complete the following steps:

1. Click **Tools** > **Internet Options** > **Accessibility**.
2. Select all the check boxes in the Formatting section.

If clicking **View** > **Text Size** > **Largest** does not increase the font size, click **Ctrl +** and **Ctrl -**.

IBM and accessibility

See the IBM Human Ability and Accessibility Center for more information about the commitment that IBM has to accessibility.

Tivoli technical training

For Tivoli technical training information, refer to the following IBM Tivoli Education Web site:

http://www.ibm.com/software/tivoli/education

Support information

If you have a problem with your IBM software, you want to resolve it quickly. IBM provides the following ways for you to obtain the support you need:

Online

Go to the IBM Software Support site at http://www.ibm.com/software/support/probsub.html and follow the instructions.

IBM Support Assistant

The IBM Support Assistant (ISA) is a free local software serviceability workbench that helps you resolve questions and problems with IBM software products. The ISA provides quick access to support-related information and serviceability tools for problem determination. To install the ISA software, go to http://www.ibm.com/software/support/isa

Conventions used in this publication

This publication uses several conventions for special terms and actions and operating system-dependent commands and paths.

Typeface conventions

This publication uses the following typeface conventions:

Bold

- Lowercase commands and mixed case commands that are otherwise difficult to distinguish from surrounding text
- Interface controls (check boxes, push buttons, radio buttons, spin buttons, fields, folders, icons, list boxes, items inside list boxes, multicolumn lists, containers, menu choices, menu names, tabs, property sheets), labels (such as **Tip:** and **Operating system considerations:**)
- Keywords and parameters in text

Italic

- Citations (examples: titles of publications, diskettes, and CDs)
- Words defined in text (example: a nonswitched line is called a *point-to-point* line)
- Emphasis of words and letters (words as words example: "Use the word *that* to introduce a restrictive clause."; letters as letters example: "The LUN address must start with the letter *L*.")
- New terms in text (except in a definition list): a *view* is a frame in a workspace that contains data
- Variables and values you must provide: ... where *myname* represents....

`Monospace`

- Examples and code examples
- File names, programming keywords, and other elements that are difficult to distinguish from surrounding text
- Message text and prompts addressed to the user
- Text that the user must type
- Values for arguments or command options

Operating system-dependent variables and paths

This publication uses environment variables without platform-specific prefixes and suffixes, unless the command applies only to specific platforms. For example, the directory where the Network Manager core components are installed is represented as NCHOME.

When using the Windows command line, preface and suffix environment variables with the percentage sign **%**, and replace each forward slash (**/**) with a backslash (\) in directory paths. For example, on Windows systems, NCHOME is %NCHOME%.

On UNIX systems, preface environment variables with the dollar sign **$**. For example, on UNIX, NCHOME is $NCHOME.

The names of environment variables are not always the same in the Windows and UNIX environments. For example, %TEMP% in Windows environments is equivalent to $TMPDIR in UNIX environments. If you are using the bash shell on a Windows system, you can use the UNIX conventions.

Chapter 1. About polling the network

To poll the network, Network Manager periodically sends queries to the devices on the network. These queries determine the behavior of the devices, for example operational status, or the data in the Management Information Base (MIB) variables of the devices.

Network polling is controlled by poll policies. Poll policies consist of the following:

- Poll definitions, which define the data to retrieve.
- Poll scope, consisting of the devices to poll. The scope can also be modified at a poll definition level to filter based on device class and interface.
- Polling interval and other poll properties.

Network Manager uses the IBM Tivoli Netcool/OMNIbus SNMP trap probe and the Syslog probe to monitor the network. To run Tivoli Netcool/OMNIbus probes, use Tivoli Netcool/OMNIbus process control.

For more information about how to use Tivoli Netcool/OMNIbus process control, see the *IBM Tivoli Netcool/OMNIbus Administration Guide*.

The polling process is controlled by the ncp_poller process. The ncp_poller process stores SNMP information in the ncmonitor database; other data is stored in-memory.

Network Manager has a multiple poller mechanism to distribute the load. If the default poller cannot handle the polling demands for your network, you might need to use the multiple poller feature.

Related tasks:

"Administering multiple pollers" on page 65
If multiple pollers are needed to poll your network, you can set up Network Manager to administer the multiple poller feature. You can add pollers or remove pollers, or use a poller ID to associate a specific poller with a policy.

Related reference:

"SNMP tables for polling in the ncmonitor database" on page 213
The SNMP tables in the ncmonitor database are used by the polling engine, ncp_poller, to store information on how to access each discovered device using SNMP.

Poll policies

Poll policies contain all the properties of a network poll operation. They specify how often a device is polled, the type of polling mechanisms employed to do the polling, and the devices to be polled.

Related reference:

Appendix A, "Default poll policies," on page 173
Network Manager IP Edition provides a set of default poll policies. Use this information to familiarize yourself with these policies.

© Copyright IBM Corp. 2006, 2016

Poll policy parameters

Use this information to understand the parameters of a poll policy.

Use the poll policy to define the following parameters:

- Name of the poll policy
- Enablement or disablement: A poll policy must be enabled for polling to take place.
- Poll definitions: A poll policy can have one or more poll definitions associated with it. If interface-level filtering is required, the poll definition must contain certain settings. For each poll definition associated with the policy, you can specify whether to store polled data for historical reporting. If this parameter is set, the data is stored in the ncpolldata database schema.

  Restriction: Storage of polled data is not supported for the Cisco Remote Ping, the Juniper Remote Ping, and the Generic Threshold poll definitions.
- Polling interval
- Poller to which to assign the poll policy, if the multiple poller feature is set up.
- Scope. This contains:
  - Network views: Specify the network views containing the devices that you wish to poll.
  - Device filters: Refine the list of devices that you want to be polled by filtering against the values of fields in the `mainNodeDetails` table of the Network Connectivity and Inventory Model (NCIM) database. Multiple filters can be combined in a Boolean relationship.

Network Manager IP Edition provides default poll policies and definitions. You might have other polls available if you have migrated poll settings during the installation process of Network Manager IP Edition.

Poll policy scope

The poll policy scope defines the devices or device interfaces to be polled.

A poll policy scope can be described as a series of filters. If, at any stage, a filter is not defined, then all devices pass through. The output of this set of filters can be either a set of devices, or, if the interface filter is defined, a set of devices interfaces. This is illustrated in the following figure.

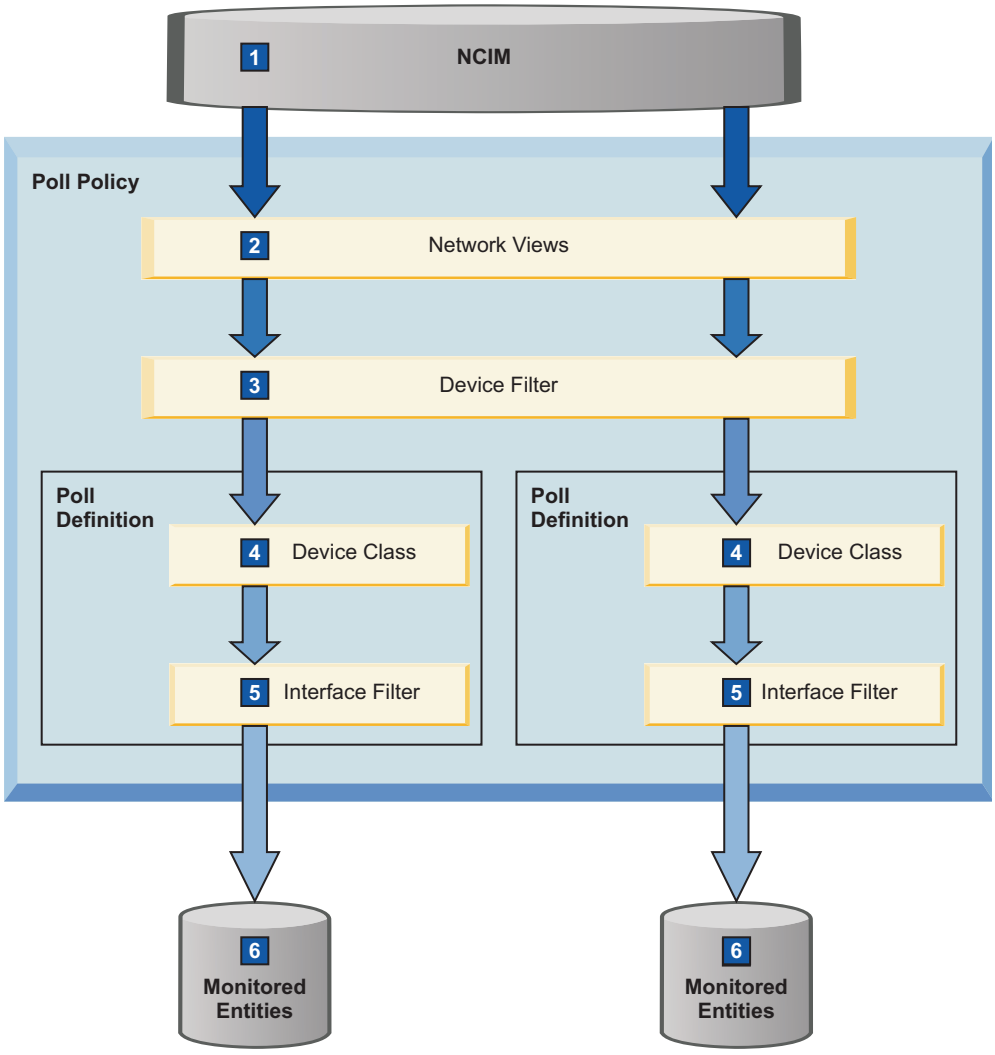


Figure 1. Poll policy scope

1 NCIM

Start with all devices defined for a single domain in the NCIM topology database.

2 Network Views

If there are any network views associated with the poll policy, then the policy scope is restricted to the devices contained by those views. If no network views are associated to the poll policy, then all devices pass through this stage. This second situation is equivalent to selecting the **All Devices** option in the **Network Views** tab of the Poll Policy Editor and the Poll Policy Wizard.

3 Device Filter

If there is a device filter defined for the poll policy, then the policy scope is further restricted to the set of devices matching the filter. If no device filter is defined then all devices that passed the Network Views filter pass through this stage. At this point, there is a set of devices available that are in scope for this poll policy. For each poll definition assigned to the policy there can be a different set of network entities in scope based on further filtering.

4 Device Class

For each poll definition assigned to the policy, the device class restricts the devices in scope based on the class selection. If no device classes are selected then no filtering occurs.

5 Interface Filter

If an interface filter is defined, and assuming that this interface filter is valid for the poll in question, then this interface filter is applied to all interfaces contained by the devices that have passed the filters above. The output is a set of in-scope interfaces. If no interface filter is defined then the output is the set of devices that passed the Device Class filter.

Poll definitions

Poll definitions determine how to poll a network entity. You must associate each poll policy with at least one poll definition. A poll policy can be associated with multiple poll definitions.

Related reference:

Appendix B, "Default poll definitions," on page 179
Network Manager IP Edition provides a number of default poll definitions that fulfil the most common polling requirements.

Poll definition parameters

Use this information to understand the parameters of a poll definition.

Network Manager provides default poll policies and definitions.

Use the poll definition to define the following parameters:

- Poll definition name
- Poll definition type: This determines the *polling mechanism* that the poll definition uses. The following polling mechanisms are used:
  - Ping polling
  - SNMP polling
- Severity level of the event that is generated

  Important: The severity level must correspond to a valid severity level as defined in IBM Tivoli Netcool/OMNIbus. For a listing of available severity levels, refer to the *IBM Tivoli Network Manager IP Edition Network Troubleshooting Guide*.
- Short description of the poll definition.
- Basic threshold poll definition type only: data label. A label used to associate the data collected by a basic threshold poll definition with a report. When defining the report, you pick the data to present in the report using the data label. This enables a report to present data tagged with the same data label but collected by more than one poll definition. For a listing of summary reports, refer to the *IBM Tivoli Network Manager IP Edition Administration Guide*.
- Scope of the poll definition: optionally which device class and interface filters to apply.
- Threshold poll definitions only: the threshold settings for generating and clearing an alert.

Polling mechanisms

Poll definitions use one of two possible polling mechanisms: ping polling and SNMP polling. All poll definitions are based on either of these mechanisms.

Ping polling

Ping polling determines the availability of a network device or interface by using an ICMP echo request.

The ping process ensures that a device is still present, live, and can be contacted in the network by periodically sending an ICMP packet to an IP address and waiting for a response.

A ping poll can have the following results:

Successful
A response to the ping packets is received. No alerts are generated.

Failure
No response to the ping packets is received within the time specified in the poll definition. Alerts are raised for network entities that do not respond.

Restore
A device that was unreachable on the last ping attempt becomes reachable again. An alert is generated to clear the ping failure alert.

Ping polling can be performed on either a chassis or an interface of a device. In the case of a chassis, the ICMP packets are sent to the IP address of a main node device. The main node IP address is also associated with an interface. In the case of interfaces, the ICMP packets are sent to the IP address of each interface. Consequently, if you enable ping polling for both chassis and interfaces, the traffic on main-node IP addresses doubles.

Remember: By default, only the chassis ping poll is enabled on all devices within the discovered network topology, with the exception of end-node devices, such as desktops and printers.

SNMP polling

SNMP polling involves retrieving Management Information Base (MIB) variables from devices in order to determine faulty behavior or connection problems. Faulty devices or faulty connections are then diagnosed by applying predefined formulas to the extracted MIB variables.

The SNMP Helper is used by polling to issue SNMP requests to network devices. For information on how to configure the SNMP Helper, see the *IBM Tivoli Network Manager IP Edition Installation and Configuration Guide*.

Link state polling:

Link state polling monitors changes to the status of the ifOperStatus and ifAdminStatus interface MIB variables. If the value of these variables changes between poll intervals, an event is raised.

Fix Pack 4 You can determine the initial state of link state polls as follows:

- Check the state of existing link state events on the interface. If the interface has no existing link state events, the initial state is set to clear.

- Poll the interface by setting the **UseFirstPollForInitialState** parameter in the `$NCHOME/etc/precision/NcPollerSchema.cfg` configuration file.

Example

If the value of ifOperStatus was 1 (up) during the previous poll, and changes to 2 (down) in the current poll, an event is raised.

The following table shows the events that are generated as a result of the changes in interface status. Also, an event is generated when a poll fails to return any data. An event with a clear severity is generated when a poll to the same device later succeeds.

Table 1. Events generated by SNMP link state polling

Status of the ifAdminStatus MIB variable between poll intervals	Status of the ifOperStatus MIB variable between poll intervals	Event generated	Event Severity
Remains 1 (up)	Changes from 1 (up) to 2 (down)	`The interface has gone down.`	Minor
Remains 1 (up)	Changes from 2 (down) to 1 (up)	`The interface has come up.`	Clear
Changes from 1 (up) to 2 (down)	Changes from 2 (down) to 1 (up)	`The interface has come up, although it should be down.`	Clear
Changes from 1 (up) to 2 (down)	Remains 2 (down)	`An administrator has confirmed that the interface should be down.`	Clear
Changes from 2 (down) to 1 (up)	Changes from 1 (up) to 2 (down)	`The interface has gone down.`	Minor
Changes from 2 (down) to 1 (up)	Remains 2 (down)	`An administrator has instructed the interface to come up, but it has not.`	Minor

Related tasks:

"Configuring Link State polling" on page 65
Specify how the **ncp_poller** process determines the initial state for Link State polls when there is no existing event. The **ncp_poller** process can use the first poll to determine the initial state of Link State polls, or assume a Clear state.

Remote ping polling:

During remote ping polling, enterprise-specific device MIBs are used to verify the status of the Multi-Protocol Label Switching (MPLS) path between devices. Specific MIB modules allow a management station to initiate ping operations remotely. With SNMP remote ping operations you can monitor ping failures by using SNMP.

During remote ping poll operations, Network Manager IP Edition instructs a Provider Edge (PE) device to periodically ping the Customer Edge (CE) device to which it is attached. The result of that remote ping operation provides information about whether the route (the MPLS path) from the PE device to the CE device is available or down.

Restriction: Remote ping operations are currently available for Cisco and Juniper devices only.

For information about setting SNMP passwords, see the *IBM Tivoli Network Manager IP Edition Discovery Guide*.

Prerequisites for remote ping polling

Before remote ping polling can operate, the following prerequisites must be met:

- You must have write access to the PE device.
- The MPLS paths must have been discovered, and the data transferred to the NCIM database. In NCIM, the data must be located as follows:
  - Virtual Private Network Router Forwarding (VRF) tables must be listed in the VPNRouteForwarding table.
  - Links from PE to CE devices must be listed in the `connects` table.
- For Juniper remote ping polling, you also require access to Juniper devices through the View-Based Access Control Model (VACM).

For more information about the VPNRouteForwarding table and the `connects` table, see the *IBM Tivoli Network Manager IP Edition Topology Database Reference*.

Threshold polling:

During threshold polling, predefined formulas are applied to the selected MIB variables, and if the threshold is exceeded by the MIB variable, then an event is generated. A Clear event is generated when the value of the MIB variable either falls below the threshold-value, or falls below a different clear-value.

You can set two thresholds:

Generate threshold
: Required: An event is generated when the value of the MIB variable or variables exceeds the threshold.

Clear threshold
: Optional: A clear-event is generated when the value of the MIB variable falls below the threshold.

If you do not specify a clear-threshold, the raised event is cleared automatically when the value of the MIB variable or variables no longer exceeds the value of the generate-threshold.

Example of threshold polling

The Monitoring Administrator wants to identify all Cisco 29xx routers that have CPU usage greater than 75%. Using SNMP polling, the administrator can monitor the behavior of all Cisco 29xx routers in the network, and define that an event is generated for each of these routers when their CPU usage exceeds 75%. A clear-threshold can also be set to generate a notification when CPU usage drops below 60%; if no clear-threshold is specified, a clear-event is generated when the CPU usage no longer exceeds 75%.

Basic and generic threshold polling

Use *basic threshold polling* to apply simple formulas to the MIB variables, or for filtering the scope at device and interface level. To filter at interface level, the poll definition must be set up for interface filtering.

Use *generic threshold polling* for complex formulas, or for filtering the scope at device level only.

Poll definition types

Each poll definition is based on a poll definition type. Poll definition types can be grouped according to the polling mechanism that they use.

Based on the polling mechanisms, the poll definition type restricts the scope of the poll operation in which it is used.

Ping polling mechanism

The ping polling mechanism has the following poll definition types:

Chassis ping
Used for pinging the management interface of a network device or the main interface of an end-node.

Interface ping
Used for ping operations on interfaces within devices. An interface ping poll definition has optional interface-level filtering.

SNMP polling mechanism

The SNMP polling mechanism has the following poll definition types:

Generic threshold
Used for setting formulas to apply against MIB variables. A generic threshold poll definition consists of the following thresholds:

Trigger threshold
Required: An event is generated when the value of the MIB variable or variables exceeds the threshold.

Clear threshold
Optional: A Clear event is generated when the value of the MIB variable falls below the threshold.

Basic threshold
Use a basic threshold to collect poll data for a single MIB variable or expression. You can present the data collected in reports or display it in MIB graphs. An event is generated when the trigger threshold condition defined in the poll definition is met, and is cleared when the clear threshold condition is met.

SNMP Link state
Used for checking the administrative and operational status. An SNMP link state poll definition has optional interface-level filtering.

Cisco remote ping
Used for checking the availability of devices by using Cisco-specific MIBs.

Juniper remote ping
For checking the availability of devices by using Juniper-specific MIBs.

Data labels

Data labels are a mechanism that allows grouping of multiple poll definitions that collect the same poll data within a single report. Data labels are only available in basic threshold poll definitions. By default the data label takes the same name as the poll definition but you can change this to meet your data labeling needs.

The following examples describe the use of data labels to enable a single report to retrieve data from multiple poll definitions. A number of Network Manager summary reports use data labels by default. For a listing of summary reports, refer to the *IBM Tivoli Network Manager IP Edition Administration Guide*.

Multiple vendor-specific poll definitions

A summary report that presents data on the percentage usage of memory across different vendor devices must retrieve poll data from multiple vendor-specific poll definitions. By defining a common `memoryPercentageUsage` data label within each of the vendor-specific poll definitions, the data retrieved by each of these different poll definitions can be grouped within one report.

Poll definitions with different thresholds and event severities

A summary report that presents data on inbound discards on device interfaces retrieves data from multiple poll definitions. Each of these poll definitions collects the same poll data but applies different thresholds and event severities to this data. By defining a common `ifInDiscards` data label within each of the different poll definitions, the data retrieved by each of these poll definitions can be grouped within a single report.

Default report to data label mapping

The following table lists the summary reports and the data labels used by default by these summary reports.

Table 2. Default report to data label mapping

Report	Data label
Router Health Summary	memoryPercentageUsage
	cpuBusy
Device Ingress Traffic Health Summary	snmpInBandwidth
	ifInDiscards
	ifInErrors
Device Egress Traffic Health Summary	snmpOutBandwidth
	ifOutDiscards
	ifOutErrors
Device Availability Summary	systemUptime

Ping polling properties and metrics

For chassis and interface ping polls, you can specify ping properties such as timeout periods and number of ping retries. You can also collect ping metrics, such as response time and packet loss.

You can specify the following ping properties when creating a chassis or interface ping poll.

Timeout
Specify, in milliseconds, how long the polling process should wait for a response from the target device before sending a new ping packet.

Retries
Specify how many times the polling process should attempt to ping the target device before giving up. When **Packet Loss** metric collection is enabled, the polling process sends this number of ping packets regardless of whether a response is received.

Payload size
Select the size of the ICMP packet to be used for the ping request. Select the default (32 bytes) or choose a custom size. This setting overrides the value of IcmpData in the `NcPollerSchema.cfg` configuration file.

CAUTION:
Using a size smaller than 32 bytes may result in packets being dropped.

You can collect the following ping metrics when creating a chassis or interface ping poll.

Response time
You can opt to collect data on the round trip time for ping tests. This is measured in milliseconds. When **Packet Loss** is also being collected, this is the average response time for each successful test.

Packet loss
You can opt to collect data on the number of ping packets for which the polling process did not receive a response. This is stored as a percentage.

Multibyte data in poll definitions

If you are running Network Manager in a domain that uses multibyte characters such as Simplified Chinese, then you must ensure that Network Manager is configured to handle multibyte characters before you configure basic or generic threshold poll definitions.

For information on how to configure Network Manager to use multibyte characters, see the *IBM Tivoli Network Manager IP Edition Installation and Configuration Guide*.

Chapter 2. Enabling and disabling polls

To activate Network Manager polling, you must enable the poll policies. If a network entity is off the network, disable the poll policy that polls that entity.

Tip: You can change the settings for a poll before enabling it. When creating your own poll policies, use the default poll policies as examples.

By default, only the chassis ping poll is enabled on all devices within the discovered network topology, with the exception of end-node devices, such as desktops and printers.

Note: If you are enabling poll policies for a large number of devices, it is best practice to wait until the poll policies are fully enabled before using the Network Polling GUI to make any changes to the poll policies. Any changes to poll policies causes the Polling engine, `ncp_poller`, to restart, and this can have unpredictable results if `ncp_poller` was in the process of enabling poll policies. Use the **Status** and **Enabled** columns in the Configure Poll Policies section of the Network Polling GUI to determine if a poll policy has been enabled.

To enable or disable polls:

1. Click **Administration** > **Network** > **Network Polling**.
2. Select the check box next to the required policy or policies.
3. Optional: To enable the selected policy or policies, click **Enable Selected Policies** .
4. Optional: To disable policies, click **Disable Selected Policies** .
5. Click **OK**.

© Copyright IBM Corp. 2006, 2016

Chapter 3. Creating polls

Create polls if the existing default poll policies and definitions do not meet your requirements. Either customize a copy of an existing or default poll, or create a new poll from scratch.

Use the Poll Policy Editor to create a fully-featured poll policy with multiple poll definitions and complete scoping facilities. Alternatively you can use the Poll Policy Wizard to guide you through the creation of a poll policy; however, you can only use the wizard to create a simple poll policy with a single existing poll definition and limited scoping facilities.

Remember: The system enforces unique poll policy names within a domain.

Note: If you are enabling poll policies for a large number of devices, it is best practice to wait until the poll policies are fully enabled before using the Network Polling GUI to make any changes to the poll policies. Any changes to poll policies causes the Polling engine, `ncp_poller`, to restart, and this can have unpredictable results if `ncp_poller` was in the process of enabling poll policies. Use the **Status** and **Enabled** columns in the Configure Poll Policies section of the Network Polling GUI to determine if a poll policy has been enabled.

Related concepts:

"Poll definition types" on page 8
Each poll definition is based on a poll definition type. Poll definition types can be grouped according to the polling mechanism that they use.

Related tasks:

Chapter 5, "Changing polls," on page 31
To change a poll, make changes to either the poll policy, or the poll definition on which the poll is based.

"Changing poll definitions" on page 36
Change existing poll definitions to customize them for your polling requirements. You change poll definitions in the Poll Definition Editor; the steps you follow differ depending on the *poll definition type*.

"Creating adaptive polls" on page 56
Create adaptive polls to enable the system to dynamically react to events on the network.

Creating fully featured poll policies

Use the Poll Policy Editor to create a fully-featured poll policy with multiple poll definitions and complete scoping features.

Using the Poll Policy Editor you can create a poll policy with the following features:

- *Multiple poll definitions*. You can use existing poll definitions or you can create new poll definitions.
- *Network views*. You can restrict the set of devices to poll to those contained by the selected network views.
- *Device Filter*. You can further refine the list of devices selected by the network views using this simple filter on the mainNodeDetails table.

© Copyright IBM Corp. 2006, 2016

Note: You can further restrict the poll policy scope by filtering the scope of each of the poll definitions contained within this poll policy. You can filter poll definition scope by device class and by interface.

1. Click **Administration** > **Network** > **Network Polling**.
2. Create a new policy by doing one of the following:
   - To create a new policy from scratch, click **Add New** . The Poll Policy Editor opens.
   - To clone an existing policy, perform the following steps:
     a. In the **Select** column select the check box next to the required row and click **Copy Selected Items** .
     b. Click **OK**. The copy is named using the following convention: *policyname*_1, where *policyname* is the name of the copied policy. For example, if you copied the policy bgpPeerState, then the copy will be named bgpPeerState_1. Poll policies are ordered alphabetically so in this example, the copy bgpPeerState_1 would appear in the list immediately after the copied policy bgpPeerState.
     c. Click the name of the copy of the poll policy in the list to open the Poll Policy Editor.
3. From the **Poll Policy Properties** tab, specify a value for each property:

   Name Type the unique name that you want to give the poll policy. Only alphanumeric characters, spaces and underscores are allowed.

   Poll Enabled
   Select this check box to enable the poll policy.

   Poll Definitions
   Use this table to specify one or more poll definitions for the poll policy.

   Refresh
   Refreshes the data in the table. This updates the table with any changes made by other users since you logged on or since you last clicked **Refresh**.

   Delete Selected Item(s)
   Deletes the selected rows.

   Add Poll Definition(s) to this Policy
   Opens the Poll Definitions panel where you can specify one or more poll definitions to add to the poll policy.

   Search
   Searches the table for text entered in the **Search** field. By default the search is performed on all of the columns in the table. Click the down arrow to the left of the **Search** field to limit the search to one or more columns in the table.
   - Select the checkboxes corresponding to the columns that you want to limit the search to.
   - Select **All Columns** to revert to the default search settings.
   - Click **OK** once you have made your selection.

Poll Definitions table

The list of poll definitions attached to this poll policy is presented in a table. You can perform the following actions on this table. Any settings made are valid for this session only.

Hide Toolbar

Hides the toolbar. If the toolbar is hidden, click Show Toolbar to show the toolbar.

Sort Column

Click the column header to sort that column in descending order. Click the column a second time to sort the column in ascending order. Further clicks toggle the column between descending and ascending order. The meaning of ascending and descending order varies according to the type of data in the column:

Alphabetical data

Ascending order orders the data from a to z. Descending order orders the data from z to a.

Numerical data

Ascending order orders the data from lowest to highest. Descending order orders the data from higest to lowest.

Icon Ascending order orders the icons from the highest to lowest value associated with the icon. Descending order orders the icons from the lowest to highest value associated with the icon. The values associated with each icon are listed below.

Resize a column

Click and drag the vertical line separator to the right of the column heading.

Select All/Clear All

Select the check box to select all rows. If all rows are selected, clear the check box to clear all rows. Select the check box next to a row to select a single row or to clear a single selected row.

Store? Select the check box to store data collected by this poll definition for reporting and historical MIB graphing purposes.

Note: This option is only available for poll definitions of type Basic Threshold.

Name The name of a poll definition attached to this poll policy. Click the name to edit the properties of this poll definition.

Type The type of poll definition.

Status Indicates whether the poll definition is in error. The full list of values is provided in the following table.

Table 3. Poll definition status

State	Value	Icon	Description
Unknown	-1		The status is unknown because the poll definition has not been run yet.
No error	0		No error. Poll definition has been run without error.
Error	Greater than 0		There is an error in the poll definition. The poll definition cannot be run. The error must be fixed before the poll definition can be used. Hover over the status icon for a pop-up with an indication of the error.

Poll Interval
Specify the required interval in seconds between poll operations. Click the arrows to change the value.

Description
Description of the poll definition.

Assign to Poller Instance
For the multiple poller feature only: Select the poller on which to run the poll policy. If only a single poller is defined, the list is read-only.

Policy Throttle
The number of devices in certain types of network views, especially event-based network views, can fluctuate and become large. In order to prevent the Polling engine, ncp_poller, from become overloaded by large numbers of devices in the network views attached to a policy, you can place a limit on the number of devices attached to a poll policy. This limit is called a policy throttle.

Specify the maximum number of entities to limit polling to. The poll policy will poll no more than the number of entities specified here.

Note: Disable policy throttling by setting this value to zero. All new poll policies have policy throttling disabled by default.

4. If you chose to add a poll definition to the poll policy then specify the poll definitions to add in the **Poll Definitions** panel using the following buttons and fields:

Refresh
Refreshes the data in the table. This updates the table with any changes made by other users since you logged on or since you last clicked **Refresh**.

Search
Searches the table for text entered in the **Search** field. By default the search is performed on all of the columns in the table. Click the down arrow to the left of the **Search** field to limit the search to one or more columns in the table.

- Select the checkboxes corresponding to the columns that you want to limit the search to.
- Select **All Columns** to revert to the default search settings.
- Click **OK** once you have made your selection.

Poll Definitions table

The complete list of poll definitions defined on the system. Poll definitions already attached to this poll policy have a greyed out checkbox. You can perform the following actions on this table. Any settings made are valid for this session only.

Hide Toolbar

Hides the toolbar. If the toolbar is hidden, click Show Toolbar to show the toolbar.

Sort Column

Click the column header to sort that column in descending order. Click the column a second time to sort the column in ascending order. Further clicks toggle the column between descending and ascending order. The meaning of ascending and descending order varies according to the type of data in the column:

Alphabetical data

Ascending order orders the data from a to z. Descending order orders the data from z to a.

Numerical data

Ascending order orders the data from lowest to highest. Descending order orders the data from higest to lowest.

Icon Ascending order orders the icons from the highest to lowest value associated with the icon. Descending order orders the icons from the lowest to highest value associated with the icon. The values associated with each icon are listed below.

Resize a column

Click and drag the vertical line separator to the right of the column heading.

Select All/Clear All

Select the check box to select all rows. If all rows are selected, clear the check box to clear all rows. Select the check box next to a row to select a single row or to clear a single selected row.

Name The name of a poll definition attached to this poll policy. Click the name to edit the properties of this poll definition.

Type The type of poll definition.

Description

Description of the poll definition.

Store Poll Data

Select this check box to store the poll data so that it can be subsequently retrieved for reporting. The data is stored in the ncpolldata database.

Restriction: Storage of polled data is not supported for the Cisco Remote Ping, the Juniper Remote Ping, and the Generic Threshold poll definitions.

Interval

Specify the required interval in seconds between poll operations. Click the arrows to change the value.

5. Click the **Network Views** tab to set the poll scope. In the **Network Views** tree, select the check boxes of the required network views. The **Network Views** tree displays only those network views that belong to the network domain in which this poll policy is defined. Cross-domain network views do not appear, because poll policies apply to only one domain.

   Attention: If you select the **All Devices** option, then the system polls all devices that match the scope defined in the **Device Filter** tab. If no scope is set then, if you select the **All Devices** option, the poll that you create will poll all devices in the current network domain.

   You can further filter the poll policy scope by filtering the scope of each of the poll definitions contained within this poll policy. You can filter poll definition scope by device class and by interface.
6. Optional: Click the **Device Filter** tab. This filters on devices on the mainNodeDetails device table only. Define the filter by using one of the following methods:
   - Type an SQL WHERE statement in the field in the Filter column.

     Note: SQL syntax is different for different databases. Refer to the documentation for the topology database that you are using for the correct SQL syntax.
   - Click **Edit** to set up the filter by using the Filter Builder.
7. Optional: In the Filter Builder, build the required query on one of the two tabs and then click **OK**:
   - On the **Basic** tab, select a field, a comparator, and type a value. Use the % character as a wildcard. The field is restricted to the selected attribute table.
   - On the **Advanced** tab, type the required SQL WHERE statement.

     Note: SQL syntax is different for different databases. Refer to the documentation for the topology database that you are using for the correct SQL syntax.

   The information that you enter on the **Basic** tab is automatically written to the **Advanced** tab.
8. Optional: To add filters on other attribute tables, click **Add new row** , and repeat the steps to edit the row and build the filter.
9. Optional: To combine multiple filters, click **All** or **Any**:
   - **All**: Only network entities that match all the specified filters are polled. For example, if you create two filters, a network entity must match both filters.
   - **Any**: Network entities that match any of the specified filters are polled.
10. Click **Save**.

Related concepts:

"Poll policy scope" on page 2
The poll policy scope defines the devices or device interfaces to be polled.

Related tasks:

"Adjusting polling bandwidth" on page 62
You can configure how much data is transferred by the Polling engine, ncp_poller, and how often. You might want to adjust polling bandwidth to avoid network congestion or to reduce the impact of large numbers of polling events occurring simultaneously.

Related reference:

Appendix D, “Syntax for poll definition expressions,” on page 189
Use this information to understand how to build complex threshold expressions to use in basic and generic threshold poll definitions.

Creating simple poll policies

Use the Poll Policy Wizard to guide you through the creation of a poll policy; however, you can only use the wizard to create a simple poll policy with a single existing poll definition and limited scoping features.

Using the Poll Policy Wizard you can create a simple poll policy with the following limited poll definition and scoping features.

- *Single poll definition*. You can use a single existing poll definition only.
- *Network views*. You can restrict the set of devices to poll to those contained by the selected network views.

  Restriction: The Poll Policy Wizard does not provide a device filter to refine the list of devices selected by the network views.

If you require a fully-featured poll policy with multiple poll definitions and full scoping features, then use the Poll Policy Editor.

1. Click **Administration** > **Network** > **Network Polling**.
2. Click **Launch Poll Configuration Wizard** .
3. Click **Next**. Complete the Poll Policy Details page as follows:

   Name Specify a name for the poll policy. Only alphanumeric characters, spaces and underscores are allowed.

   Interval
   Specify the required interval in seconds between poll operations. Click the arrows to change the value.

   Poll Enabled
   Specify whether the poll should be enabled. The poll is enabled by default. To disable the poll, clear this check box.

   Store Poll Data
   Select this check box to store the poll data so that it can be subsequently retrieved for reporting. The data is stored in the ncpolldata database.

   Restriction: Storage of polled data is not supported for the Cisco Remote Ping, the Juniper Remote Ping, and the Generic Threshold poll definitions.

   Definition
   Select a poll definition from the list.
4. Click **Next**. On the Poll Policy Scope Details page, select the check boxes of the required network views. In the **Network Views** tree, select the check boxes of the required network views. The **Network Views** tree displays only those network views that belong to the network domain in which this poll policy is defined. Cross-domain network views do not appear, because poll policies apply to only one domain.

   Attention: If you select the **All Devices** option, the poll that you create will poll all devices in the current network domain.

5. Click **Next**. On the Poll Policy Summary page, review the information that you specified and click **Finish**.

Related concepts:

"Poll policy scope" on page 2
The poll policy scope defines the devices or device interfaces to be polled.

Related reference:

Appendix D, "Syntax for poll definition expressions," on page 189
Use this information to understand how to build complex threshold expressions to use in basic and generic threshold poll definitions.

Quick reference for poll policy creation based on custom data

You can create a poll policy that polls the network based on custom device data collected as part of a discovery. Do this by configuring the poll policy scope to be a network view organized by custom data. For example, you can configure the discovery to collect custom data such as the customer name and customer type associated with interfaces on a device. You can then create network views based on this custom data; for example, you could create network views of devices organized by customer name. You can then set your poll policy scope to poll specified customer network views.

The steps are described in the following table.

Table 4. Quick reference for poll policy creation based on custom data

Action	Further information
1. Configure discovery to tag devices or interfaces with custom data. For example, you can configure the discovery to tag devices or interfaces with name-value pair tags such as the customer name and the customer location associated with that interface or device. You can do this using one of the methods shown to the right.	If you are using the File finder to seed your discovery, then you can add name-value pair tags to entities by adding extra columns to the seed file read by the File finder. For further information, see the *IBM Tivoli Network Manager IP Edition Discovery Guide*.
	If you are using the Ping finder to seed your discovery, then you can assign name-value pair tags to interfaces or devices using one of the following methods: • Using unique IP addresses. • Using a filtered set of IP addresses.. • If you choose the filter method. then you can use the GetCustomTag stitcher to use logic to evaluate the value part of the name-value pair. For further information, see the *IBM Tivoli Network Manager IP Edition Discovery Guide*.
2. Extend the NCIM topology database to store the custom data.	For example, you can add name-value pair tags to the NCIM entityDetails table. For further information, see the *IBM Tivoli Network Manager IP Edition Topology Database Reference*.
	If you have configured discovery to collect multiple custom data fields for each device or interface, then you can create a new NCIM table to store this custom data. For further information, see the *IBM Tivoli Network Manager IP Edition Topology Database Reference*.

Table 4. Quick reference for poll policy creation based on custom data (continued)

Action	Further information
3. Create a network view or browse the Hop View to display the parts of the network associated with the custom data.	You can create a filtered network view or a dynamic distinct network view to display the parts of the network related to the custom data. • Create a filtered network view to display parts of the network based on topology database filters that use the custom data. For example you could create a filtered network view that uses the NCIM entityDetails table to display all devices containing interfaces tagged with a specified customer name. • Create a distinct dynamic network view to display parts of the network based on categories and subcategories in the custom data. For example, you could create a distinct dynamic view that uses the NCIM entityDetails table to generate a set of customer views, each view corresponding to one of the customer names in the custom data. Each customer view would display all devices containing interfaces tagged with that customer name. For further information, see the *IBM Tivoli Network Manager IP Edition Network Visualization Setup Guide*.
	You can use the advanced search in the Hop View to find a device based on attributes within the custom data. For example, you could use the Entity Search window within the Hop View to browse the entityDetails table and retrieve all devices for customers in a specific location. For further information, see the *IBM Tivoli Network Manager IP Edition Network Troubleshooting Guide*.
4. Create a poll policy that uses the network view created in the previous step as the policy scope.	As you create the poll policy you are provided the option to specify which network views to poll. By selecting one or more of the network views that you created in the previous step, you are able to poll devices that contain interfaces tagged with a specified customer name. For further information, see the *IBM Tivoli Network Manager IP Edition Event Management Guide*.

Chapter 4. Creating new poll definitions

Use the Poll Definition Editor to guide you through the steps of creating a new poll definition.

Before you create or change a poll definition, view an existing poll definition to determine whether you can use it as a template to create a new poll definition.

Remember: The system enforces unique poll definition names within a domain.

Because the poll definition types differ, the Poll Definition Editor displays different pages depending on which poll definition type you select.

Related tasks:

"Changing poll definitions" on page 36
Change existing poll definitions to customize them for your polling requirements. You change poll definitions in the Poll Definition Editor; the steps you follow differ depending on the *poll definition type*.

Chapter 3, "Creating polls," on page 13
Create polls if the existing default poll policies and definitions do not meet your requirements. Either customize a copy of an existing or default poll, or create a new poll from scratch.

Related reference:

Appendix B, "Default poll definitions," on page 179
Network Manager IP Edition provides a number of default poll definitions that fulfil the most common polling requirements.

Creating basic threshold poll definitions

Create a basic threshold poll definition to run simple formulas against MIB variables, or to create threshold polls with interface-level filtering.

Before you create or change a poll definition, view an existing poll definition to determine whether you can use it as a template to create a new poll definition.

To create a basic threshold poll definition:

1. Click **Administration** > **Network** > **Network Polling**.
2. Click **Add New** . The New Poll Definition Type Selection page is displayed.
3. Select `Basic threshold` from the list and click **OK**.
4. In the Poll Definition Editor, under the **General** tab, complete the **General Properties** fields as follows:

   Name Specify a unique name for the poll definition. Only alphanumeric characters, spaces and underscores are allowed.

   Type This field is disabled. The Polling engine, ncp_poller, automatically populates this field once this poll definition is included as part of an enabled poll policy. The value assigned to the event ID is POLL-pollde

   Event ID
   This field is disabled. The Polling engine, ncp_poller, automatically

© Copyright IBM Corp. 2006, 2016

populates this field once this poll definition is included as part of an enabled policy. The **Event ID** field is populated as follows:

- If this is a new poll definition, then the **Event ID** field is populated with the value `POLL-polldef`, where *polldef* is the name of the current poll definition.
- If you created a poll definition by copying an existing poll definition, then the **Event ID** contains the same value as the copied poll definition.

Note: Some of the older default polls have **Event ID** fields that do not use the `POLL-polldef` naming convention.

Event Severity
Specify a valid number for the severity. The severity level must correspond to a valid severity level as defined in IBM Tivoli Netcool/OMNIbus. For a listing of available severity levels, refer to the *IBM Tivoli Network Manager IP Edition Network Troubleshooting Guide*

Description
Type a short description of the poll definition.

Data Label
Click the data label list and select one of the data labels from the list. By default the data label takes the same name as the current poll definition. To define a new data label, select `<Add New Data Label>`. The field to the right of the list becomes active. Type the name of the new data label in this field.

5. Click the **Classes** tab. In the **Classes** tree, select the check boxes of the required classes.

   Attention: If you leave all classes unchecked, then the system polls all devices that match the scope defined in the poll policy that uses this poll definition.
6. Optional: Click the **Interface Filter** tab and build the filter against the required fields. The **Table** field is prepopulated with the `interfaces` table.
7. Click the **Poll Data** tab and specify the required formula:
   - To specify a MIB Object Identifier (OID), select **Single OID**. Specify the current or delta value of the required MIB variable and type the variable into the next field.
   - To specify a complex expression, select **Expression** and type the formula into the field.

   To select variables directly from the MIB tree, click **Add MIB Object** . From the MIB tree, you can specify the current or previous values of the selected MIB variable, or resolve the current value of the variable to the SNMP index.
8. Click the **Threshold** tab and specify the formulas for triggering events and clearing events. The MIB OID or expression that you specified on the **Poll Data** tab is written automatically into the formulas.
   a. In the **Trigger Threshold** area, select a comparator from the list and type the value against which to filter the MIB OID.
   b. In the **Description** field, type a meaningful description of the trigger formula. Add the MIB variable to the description in parentheses. The description is displayed in the AEL when and event is raised. For example:
      `CPU usage high (avgBusy5=)`
   c. To insert the underlying eval statement into the description, position the cursor before the closing parenthesis, click **Add MIB Object** , and

navigate to the specified variable. Specify whether the current or previous value of the variable is evaluated, or whether the value is resolved to the SNMP index, and click **OK**. The statement is inserted, for example:

```
CPU usage high (avgBusy5=eval(text,"&SNMP.VALUE.sysName"))
```

d. Repeat steps 8a on page 24 to 8c on page 24 for the **Clear Threshold** area.

9. Click **Save**.

Related concepts:

"Multibyte data in poll definitions" on page 10
If you are running Network Manager in a domain that uses multibyte characters such as Simplified Chinese, then you must ensure that Network Manager is configured to handle multibyte characters before you configure basic or generic threshold poll definitions.

"Poll policy scope" on page 2
The poll policy scope defines the devices or device interfaces to be polled.

Related reference:

"Example basic threshold expression" on page 44
Use this example basic threshold expression to understand how to compose complex basic threshold expressions.

Appendix D, "Syntax for poll definition expressions," on page 189
Use this information to understand how to build complex threshold expressions to use in basic and generic threshold poll definitions.

Creating generic threshold poll definitions

Use the Poll Definition Editor to create new generic threshold poll definitions.

When creating a generic threshold definition, you set formulas and combine formulas.

To create a generic threshold poll definition:

1. Click **Administration** > **Network** > **Network Polling**.
2. Click **Add New** . The New Poll Definition Type Selection page is displayed.
3. Select `Generic Threshold` from the list and click **OK**.
4. In the Poll Definition Editor, under the **General** tab, complete the **General Properties** fields as follows:

   Name Specify a unique name for the poll definition. Only alphanumeric characters, spaces and underscores are allowed.

   Type This field is disabled. The Polling engine, ncp_poller, automatically populates this field once this poll definition is included as part of an enabled poll policy. The value assigned to the event ID is POLL-pollde

   Event ID
   This field is disabled. The Polling engine, ncp_poller, automatically populates this field once this poll definition is included as part of an enabled policy. The **Event ID** field is populated as follows:

   - If this is a new poll definition, then the **Event ID** field is populated with the value `POLL-`*`polldef`*, where *`polldef`* is the name of the current poll definition.

- If you created a poll definition by copying an existing poll definition, then the **Event ID** contains the same value as the copied poll definition.

Note: Some of the older default polls have **Event ID** fields that do not use the `POLL-polldef` naming convention.

Event Severity
Specify a valid number for the severity. The severity level must correspond to a valid severity level as defined in IBM Tivoli Netcool/OMNIbus. For a listing of available severity levels, refer to the *IBM Tivoli Network Manager IP Edition Network Troubleshooting Guide*

Description
Type a short description of the poll definition.

5. Click the **Classes** tab. In the **Classes** tree, select the check boxes of the required classes.

   Attention: If you leave all classes unchecked, then the system polls all devices that match the scope defined in the poll policy that uses this poll definition.
6. Optional: Click the **Interface Filter** tab and build the filter against the required fields. The **Table** field is prepopulated with the `interfaces` table.
7. Click the **Trigger Threshold** tab. Build the formula that specifies the threshold by using one of the following methods:
   - In the **Basic** area, use the fields and options to build a formula. To select values from the MIB tree, click **Open MIB Tree** .
   - In the **Advanced** area, type the required **`eval`** statement in Object Query Language (OQL).
8. Specify the message that is displayed in the AEL for the generated event:
   a. In the **Event description** field, type the message.
   b. To insert the MIB variables in the field, click **Open MIB Tree** . Set the message to include either the current or previous SNMP value, or the SNMP index, and click **OK**.
9. Required: Click the **Clear Threshold** tab. Every generic threshold poll definition requires a clear threshold. Build the formula that specifies the threshold by using one of the following methods:
   - In the **Basic** area, use the fields and options to build a formula. To select values from the MIB tree, click **Open MIB Tree** .
   - In the **Advanced** area, type the required **`eval`** statement in Object Query Language (OQL).

   Tip: If you want the threshold to be cleared manually, create a clear threshold that will not be reached.
10. Specify the message that is displayed in the AEL for the generated event:
    a. In the **Event description** field, type the message.
    b. To insert the MIB variables in the field, click **Open MIB Tree** . Set the message to include either the current or previous SNMP value, or the SNMP index, and click **OK**.
11. Click **Save**, then click **OK**.

The poll definition is added to the bottom of the list.

Related concepts:

"Multibyte data in poll definitions" on page 10
If you are running Network Manager in a domain that uses multibyte characters such as Simplified Chinese, then you must ensure that Network Manager is configured to handle multibyte characters before you configure basic or generic threshold poll definitions.

"Poll policy scope" on page 2
The poll policy scope defines the devices or device interfaces to be polled.

Related reference:

"Example generic threshold expression" on page 44
Use this example generic threshold expression to understand how to compose complex generic threshold expressions.

Appendix D, "Syntax for poll definition expressions," on page 189
Use this information to understand how to build complex threshold expressions to use in basic and generic threshold poll definitions.

Creating chassis and interface ping poll definitions

Use the Poll Definition Editor to create chassis and interface ping poll definition types.

You perform identical steps to create a poll definition based on all the above poll definition types.

To create a chassis or interface ping poll definition:

1. Click **Administration** > **Network** > **Network Polling**.
2. Click **Add New** . The New Poll Definition Type Selection page is displayed.
3. Select Chassis Ping or Interface Ping from the list.
4. In the Poll Definition Editor, under the **General** tab, complete the **General Properties** fields as follows:

   Name Specify a unique name for the poll definition. Only alphanumeric characters, spaces and underscores are allowed.

   Type This field is disabled. The Polling engine, ncp_poller, automatically populates this field once this poll definition is included as part of an enabled poll policy. The value assigned to the event ID is POLL-pollde

   Event ID
   This field is disabled. The Polling engine, ncp_poller, automatically populates this field once this poll definition is included as part of an enabled policy. The **Event ID** field is populated as follows:

   - If this is a new poll definition, then the **Event ID** field is populated with the value `POLL-`*`polldef`*, where *`polldef`* is the name of the current poll definition.
   - If you created a poll definition by copying an existing poll definition, then the **Event ID** contains the same value as the copied poll definition.

   Note: Some of the older default polls have **Event ID** fields that do not use the `POLL-`*`polldef`* naming convention.

Event Severity
Specify a valid number for the severity. The severity level must correspond to a valid severity level as defined in IBM Tivoli Netcool/OMNIbus. For a listing of available severity levels, refer to the *IBM Tivoli Network Manager IP Edition Network Troubleshooting Guide*

Description
Type a short description of the poll definition.

5. Click the **Classes** tab. In the **Classes** tree, select the check boxes of the required classes.

   Attention: If you leave all classes unchecked, then the system polls all devices that match the scope defined in the poll policy that uses this poll definition.
6. Optional: Click the **Interface Filter** tab and build the filter against the required fields. The **Table** field is prepopulated with the `interfaces` table.
7. Click the **Ping** tab and complete the **Ping Properties** fields as follows:

   Timeout
   Specify, in milliseconds, how long the polling process should wait for a response from the target device before sending a new ping packet.

   Retries
   Specify how many times the polling process should attempt to ping the target device before giving up. When **Packet Loss** metric collection is enabled, the polling process sends this number of ping packets regardless of whether a response is received.

   Collect Ping Metrics

   Response Time
   Check the box to collect the time taken by devices to respond to a ping request.

   Packet Loss
   Check the box to collect data about lost packets.

   Payload Size
   Select the size of the ICMP packet to be used for the ping request. Select the default (32 bytes) or choose a custom size. This setting overrides the value of IcmpData in the `NcPollerSchema.cfg` configuration file.

   CAUTION:
   Using a size smaller than 32 bytes may result in packets being dropped.
8. Click **Save**, then click **OK**.

Related concepts:

"Poll policy scope" on page 2
The poll policy scope defines the devices or device interfaces to be polled.

Creating remote ping and link state poll definitions

Use the Poll Definition Editor to create new poll definitions with the following poll definition types: Cisco remote ping, Juniper remote ping, SNMP link state.

You perform identical steps to create a poll definition based on all the above poll definition types.

To create a remote ping poll definition, or an SNMP link state poll definition:

1. Click **Administration** > **Network** > **Network Polling**.
2. Click **Add New** . The New Poll Definition Type Selection page is displayed.
3. Select the required definition type from the list:
   - `Cisco Remote Ping`
   - `Juniper Remote Ping`
   - `SNMP Link State`
4. Click **OK**.
5. In the Poll Definition Editor, under the **General** tab, complete the **General Properties** fields as follows:

   Name Specify a unique name for the poll definition. Only alphanumeric characters, spaces and underscores are allowed.

   Type This field is disabled. The Polling engine, ncp_poller, automatically populates this field once this poll definition is included as part of an enabled poll policy. The value assigned to the event ID is POLL-pollde

   Event ID
   This field is disabled. The Polling engine, ncp_poller, automatically populates this field once this poll definition is included as part of an enabled policy. The **Event ID** field is populated as follows:
   - If this is a new poll definition, then the **Event ID** field is populated with the value `POLL-`*`polldef`*, where *`polldef`* is the name of the current poll definition.
   - If you created a poll definition by copying an existing poll definition, then the **Event ID** contains the same value as the copied poll definition.

   Note: Some of the older default polls have **Event ID** fields that do not use the `POLL-`*`polldef`* naming convention.

   Event Severity
   Specify a valid number for the severity. The severity level must correspond to a valid severity level as defined in IBM Tivoli Netcool/OMNIbus. For a listing of available severity levels, refer to the *IBM Tivoli Network Manager IP Edition Network Troubleshooting Guide*

   Description
   Type a short description of the poll definition.
6. Click the **Classes** tab. In the **Classes** tree, select the check boxes of the required classes.

   Attention: If you leave all classes unchecked, then the system polls all devices that match the scope defined in the poll policy that uses this poll definition.

7. Optional: Click the **Interface Filter** tab and build the filter against the required fields. The **Table** field is prepopulated with the `interfaces` table.
8. Click **Save**, then click **OK**.

Related concepts:

"Poll policy scope" on page 2
The poll policy scope defines the devices or device interfaces to be polled.

Chapter 5. Changing polls

To change a poll, make changes to either the poll policy, or the poll definition on which the poll is based.

Related concepts:

"Poll policies" on page 1
Poll policies contain all the properties of a network poll operation. They specify how often a device is polled, the type of polling mechanisms employed to do the polling, and the devices to be polled.

"Poll definitions" on page 4
Poll definitions determine how to poll a network entity. You must associate each poll policy with at least one poll definition. A poll policy can be associated with multiple poll definitions.

Related tasks:

Chapter 3, "Creating polls," on page 13
Create polls if the existing default poll policies and definitions do not meet your requirements. Either customize a copy of an existing or default poll, or create a new poll from scratch.

Related reference:

Appendix A, "Default poll policies," on page 173
Network Manager IP Edition provides a set of default poll policies. Use this information to familiarize yourself with these policies.

Changing poll policies

Use the Poll Policy Editor to change the settings of existing poll policies.

Before you create or change a poll policy, view an existing poll policy to determine whether you can use the poll as a template to create a new poll policy.

Note: If you are enabling poll policies for a large number of devices, it is best practice to wait until the poll policies are fully enabled before using the Network Polling GUI to make any changes to the poll policies. Any changes to poll policies causes the Polling engine, `ncp_poller`, to restart, and this can have unpredictable results if `ncp_poller` was in the process of enabling poll policies. Use the **Status** and **Enabled** columns in the Configure Poll Policies section of the Network Polling GUI to determine if a poll policy has been enabled.

To change a poll policy:

1. Click **Administration** > **Network** > **Network Polling**.
2. Click the required poll policy. The Poll Policy Editor is displayed; the settings of the selected poll policy are automatically loaded into the fields.
3. Under **Poll Policy Properties**, specify a value for the following fields:

   Name Type the unique name that you want to give the poll policy. Only alphanumeric characters, spaces and underscores are allowed.

   Poll Enabled
   Select this check box to enable the poll policy.

   Poll Definitions
   Use this table to specify one or more poll definitions for the poll policy.

© Copyright IBM Corp. 2006, 2016

Refresh

Refreshes the data in the table. This updates the table with any changes made by other users since you logged on or since you last clicked **Refresh**.

Delete Selected Item(s)

Deletes the selected rows.

Add Poll Definition(s) to this Policy

Opens the Poll Definitions panel where you can specify one or more poll definitions to add to the poll policy.

Search

Searches the table for text entered in the **Search** field. By default the search is performed on all of the columns in the table. Click the down arrow to the left of the **Search** field to limit the search to one or more columns in the table.

- Select the checkboxes corresponding to the columns that you want to limit the search to.
- Select **All Columns** to revert to the default search settings.
- Click **OK** once you have made your selection.

Poll Definitions table

The list of poll definitions attached to this poll policy is presented in a table. You can perform the following actions on this table. Any settings made are valid for this session only.

Hide Toolbar

Hides the toolbar. If the toolbar is hidden, click Show Toolbar to show the toolbar.

Sort Column

Click the column header to sort that column in descending order. Click the column a second time to sort the column in ascending order. Further clicks toggle the column between descending and ascending order. The meaning of ascending and descending order varies according to the type of data in the column:

Alphabetical data

Ascending order orders the data from a to z. Descending order orders the data from z to a.

Numerical data

Ascending order orders the data from lowest to highest. Descending order orders the data from higest to lowest.

Icon Ascending order orders the icons from the highest to lowest value associated with the icon. Descending order orders the icons from the lowest to highest value associated with the icon. The values associated with each icon are listed below.

Resize a column
Click and drag the vertical line separator to the right of the column heading.

Select All/Clear All
Select the check box to select all rows. If all rows are selected, clear the check box to clear all rows. Select the check box next to a row to select a single row or to clear a single selected row.

Store? Select the check box to store data collected by this poll definition for reporting and historical MIB graphing purposes.

Note: This option is only available for poll definitions of type Basic Threshold.

Name The name of a poll definition attached to this poll policy. Click the name to edit the properties of this poll definition.

Type The type of poll definition.

Status Indicates whether the poll definition is in error. The full list of values is provided in the following table.

Table 5. Poll definition status

State	Value	Icon	Description
Unknown	-1		The status is unknown because the poll definition has not been run yet.
No error	0		No error. Poll definition has been run without error.
Error	Greater than 0		There is an error in the poll definition. The poll definition cannot be run. The error must be fixed before the poll definition can be used. Hover over the status icon for a pop-up with an indication of the error.

Poll Interval
Specify the required interval in seconds between poll operations. Click the arrows to change the value.

Description
Description of the poll definition.

Assign to Poller Instance
For the multiple poller feature only: Select the poller on which to run the poll policy. If only a single poller is defined, the list is read-only.

Policy Throttle
The number of devices in certain types of network views, especially event-based network views, can fluctuate and become large. In order to prevent the Polling engine, ncp_poller, from become overloaded by large numbers of devices in the network views attached to a policy, you can place a limit on the number of devices attached to a poll policy. This limit is called a policy throttle.

Specify the maximum number of entities to limit polling to. The poll policy will poll no more than the number of entities specified here.

Note: Disable policy throttling by setting this value to zero. All new poll policies have policy throttling disabled by default.

4. Click the **Network Views** tab. In the **Network Views** tree, select the check boxes of the required network views. The **Network Views** tree displays only those network views that belong to the network domain in which this poll policy is defined.

   Attention: If you select the **All Devices** option, then the system polls all devices that match the scope defined in the **Device Filter** tab. If no scope is set then, if you select the **All Devices** option, the poll that you create will poll all devices in the current network domain.
5. Optional: Click the **Device Filter** tab. This filters on devices on the mainNodeDetails device table only. Define the filter by using one of the following methods:
   - Type an SQL WHERE statement in the field in the Filter column.

     Note: SQL syntax is different for different databases. Refer to the documentation for the topology database that you are using for the correct SQL syntax.
   - Click **Edit** to set up the filter by using the Filter Builder.
6. In the Filter Builder, build the required query on one of the two tabs and then click **OK**:
   - On the **Basic** tab, select a field, a comparator, and type a value. Use the % character as a wildcard. The field is restricted to the selected attribute table.
   - On the **Advanced** tab, type the required SQL WHERE statement.

     Note: SQL syntax is different for different databases. Refer to the documentation for the topology database that you are using for the correct SQL syntax.

   The information that you enter on the **Basic** tab is automatically written to the **Advanced** tab.
7. To add filters on other attribute tables, click **Add new row** , and repeat the steps to edit the row and build the filter.
8. To combine multiple filters, click **All** or **Any**:
   - **All**: Only network entities that match all the specified filters are polled. For example, if you create two filters, a network entity must match both filters.
   - **Any**: Network entities that match any of the specified filters are polled.
9. Click **Save**.

Related reference:

Appendix A, “Default poll policies,” on page 173
Network Manager IP Edition provides a set of default poll policies. Use this information to familiarize yourself with these policies.

Example poll policy

Use this example of a customized poll policy to help you copy an existing policy and customize it to poll specific devices in class C subnets.

Scenario

You must customize a poll policy to meet the following requirements:

- The poll policy must check the network for devices in class C subnets that have the IP address 9.1.2.* or 10.123.46*
- The poll policy must check the network at intervals of 60 seconds
- The poll policy must begin polling the network immediately after it has been saved

Required settings

To create a poll that responds to the requirements described in the previous scenario, make the following settings:

1. On the Configure Poll Policies page, make a copy of the ciscoMemoryPctgUsage poll policy. The copy of the poll policy appears as a new row in the poll policy table.
2. Find the row containing the copy of the ciscoMemoryPctgUsage poll policy, and click the name of the copy of the poll policy. This enables you to edit the copy of the poll policy in the Poll Policy Editor.
3. On the **Poll Policy Properties** tab, make the following settings:

   Name Type a meaningful name, for example `ciscoMemoryPctgUsage for class C subnets with 9.1.2* and 10.123.46*`.

   Poll Enabled:
   Select this check box.

   Poll Interval
   In the **Poll Definitions** table, scroll across and type `60` in the **Poll Interval** column.
4. On the **Network Views** tab, ensure that **All Devices** is selected.
5. On the **Device Filter** tab, make the following settings:
   a. Select **Any**.
   b. To specify the filter against fields of the mainNodeDetails table, click **Open Filter Builder** .
   c. On the **Basic** tab of the Filter Builder, complete the fields as follows:

Field	Comparator	Value
ipAddress	like	9.1.2.%

   d. Click **OK**.
   e. Click **Add** .
   f. To specify another filter against fields of the mainNodeDetails table, click **Open Filter Builder** .
   g. On the **Basic** tab of the Filter Builder, complete the fields as follows:

Field	Comparator	Value
ipAddress	like	10.123.46.%

h. Click **OK**.

6. Click **Save**.

Related reference:

Appendix A, "Default poll policies," on page 173
Network Manager IP Edition provides a set of default poll policies. Use this information to familiarize yourself with these policies.

Changing poll definitions

Change existing poll definitions to customize them for your polling requirements. You change poll definitions in the Poll Definition Editor; the steps you follow differ depending on the *poll definition type*.

Before you create or change a poll definition, view an existing poll definition to determine whether you can use it as a template to create a new poll definition.

Related tasks:

Chapter 4, "Creating new poll definitions," on page 23
Use the Poll Definition Editor to guide you through the steps of creating a new poll definition.

Related reference:

Appendix A, "Default poll policies," on page 173
Network Manager IP Edition provides a set of default poll policies. Use this information to familiarize yourself with these policies.

Changing basic threshold poll definitions

Use the Poll Definition Editor to change basic threshold poll definitions.

You can change some of the general properties of the poll definition and the properties that are associated with the poll definition type. However, you cannot change the poll definition type.

To change a basic threshold poll definition:

1. Click **Administration** > **Network** > **Network Polling**.
2. Click the required poll definition. The poll definition must have the poll definition type Basic Threshold.
3. In the Poll Definition Editor, under the **General** tab, complete the **General Properties** fields as follows:

   Name Specify a unique name for the poll definition. Only alphanumeric characters, spaces and underscores are allowed.

   Type This field is disabled. The Polling engine, ncp_poller, automatically populates this field once this poll definition is included as part of an enabled poll policy. The value assigned to the event ID is POLL-pollde

   Event ID
   This field is disabled. The Polling engine, ncp_poller, automatically populates this field once this poll definition is included as part of an enabled policy. The **Event ID** field is populated as follows:

   - If this is a new poll definition, then the **Event ID** field is populated with the value `POLL-`*`polldef`*, where *`polldef`* is the name of the current poll definition.

- If you created a poll definition by copying an existing poll definition, then the **Event ID** contains the same value as the copied poll definition.

  Note: Some of the older default polls have **Event ID** fields that do not use the `POLL-polldef` naming convention.

Event Severity
Specify a valid number for the severity. The severity level must correspond to a valid severity level as defined in IBM Tivoli Netcool/OMNIbus. For a listing of available severity levels, refer to the *IBM Tivoli Network Manager IP Edition Network Troubleshooting Guide*

Description
Type a short description of the poll definition.

Data Label
Click the data label list and select one of the data labels from the list. By default the data label takes the same name as the current poll definition. To define a new data label, select `<Add New Data Label>`. The field to the right of the list becomes active. Type the name of the new data label in this field.

4. Click the **Classes** tab. In the **Classes** tree, select the check boxes of the required classes.

   Attention: If you leave all classes unchecked, then the system polls all devices that match the scope defined in the poll policy that uses this poll definition.
5. Optional: Click the **Interface Filter** tab and build the filter against the required fields. The **Table** field is prepopulated with the `interfaces` table.
6. Click the **Poll Data** tab and specify the required formula:
   - To specify a MIB Object Identifier (OID), select **Single OID**. Specify the current or delta value of the required MIB variable and type the variable into the next field.
   - To specify a complex expression, select **Expression** and type the formula into the field.

   To select variables directly from the MIB tree, click **Add MIB Object** . From the MIB tree, you can specify the current or previous values of the selected MIB variable, or resolve the current value of the variable to the SNMP index.
7. Click the **Threshold** tab and specify the formulas for triggering events and clearing events. The MIB OID or expression that you specified on the **Poll Data** tab is written automatically into the formulas.
   a. In the **Trigger Threshold** area, select a comparator from the list and type the value against which to filter the MIB OID.
   b. In the **Description** field, type a meaningful description of the trigger formula. Add the MIB variable to the description in parentheses. The description is displayed in the AEL when and event is raised. For example:
      ```
      CPU usage high (avgBusy5=)
      ```
   c. To insert the underlying eval statement into the description, position the cursor before the closing parenthesis, click **Add MIB Object** , and navigate to the specified variable. Specify whether the current or previous value of the variable is evaluated, or whether the value is resolved to the SNMP index, and click **OK**. The statement is inserted, for example:
      ```
      CPU usage high (avgBusy5=eval(text,"&SNMP.VALUE.sysName"))
      ```
   d. Repeat steps 7a to 7c for the **Clear Threshold** area.

8. Click **Save**.

Related concepts:

"Multibyte data in poll definitions" on page 10
If you are running Network Manager in a domain that uses multibyte characters such as Simplified Chinese, then you must ensure that Network Manager is configured to handle multibyte characters before you configure basic or generic threshold poll definitions.

"Poll policy scope" on page 2
The poll policy scope defines the devices or device interfaces to be polled.

Related reference:

"Example basic threshold expression" on page 44
Use this example basic threshold expression to understand how to compose complex basic threshold expressions.

Appendix D, "Syntax for poll definition expressions," on page 189
Use this information to understand how to build complex threshold expressions to use in basic and generic threshold poll definitions.

Changing generic threshold poll definitions

Use the Poll Definition Editor to change generic threshold poll definitions.

You can change some of the general properties of the poll definition and the properties that are associated with the poll definition type. However, you cannot change the poll definition type.

To change a generic threshold poll definition:

1. Click **Administration** > **Network** > **Network Polling**.
2. Click the required poll definition. The poll definition must have the poll definition type Generic Threshold.
3. In the Poll Definition Editor, under the **General** tab, complete the **General Properties** fields as follows:

   Name Specify a unique name for the poll definition. Only alphanumeric characters, spaces and underscores are allowed.

   Type This field is disabled. The Polling engine, ncp_poller, automatically populates this field once this poll definition is included as part of an enabled poll policy. The value assigned to the event ID is POLL-pollde

   Event ID
   This field is disabled. The Polling engine, ncp_poller, automatically populates this field once this poll definition is included as part of an enabled policy. The **Event ID** field is populated as follows:

   - If this is a new poll definition, then the **Event ID** field is populated with the value `POLL-polldef`, where `polldef` is the name of the current poll definition.
   - If you created a poll definition by copying an existing poll definition, then the **Event ID** contains the same value as the copied poll definition.

   Note: Some of the older default polls have **Event ID** fields that do not use the `POLL-polldef` naming convention.

   Event Severity
   Specify a valid number for the severity. The severity level must correspond to a valid severity level as defined in IBM Tivoli

Netcool/OMNIbus. For a listing of available severity levels, refer to the *IBM Tivoli Network Manager IP Edition Network Troubleshooting Guide*

Description
Type a short description of the poll definition.

4. Click the **Classes** tab. In the **Classes** tree, select the check boxes of the required classes.

   Attention: If you leave all classes unchecked, then the system polls all devices that match the scope defined in the poll policy that uses this poll definition.

5. Optional: Click the **Interface Filter** tab and build the filter against the required fields. The **Table** field is prepopulated with the `interfaces` table.
6. Click the **Trigger Threshold** tab. Build the formula that specifies the threshold by using one of the following methods:
   - In the **Basic** area, use the fields and options to build a formula. To select values from the MIB tree, click **Open MIB Tree** .
   - In the **Advanced** area, type the required **`eval`** statement in Object Query Language (OQL).
7. Specify the message that is displayed in the AEL for the generated event:
   a. In the **Event description** field, type the message.
   b. To insert the MIB variables in the field, click **Open MIB Tree** . Set the message to include either the current or previous SNMP value, or the SNMP index, and click **OK**.
8. Required: Click the **Clear Threshold** tab. Every generic threshold poll definition requires a clear threshold. Build the formula that specifies the threshold by using one of the following methods:
   - In the **Basic** area, use the fields and options to build a formula. To select values from the MIB tree, click **Open MIB Tree** .
   - In the **Advanced** area, type the required **`eval`** statement in Object Query Language (OQL).

   Tip: If you want the threshold to be cleared manually, create a clear threshold that will not be reached.
9. Specify the message that is displayed in the AEL for the generated event:
   a. In the **Event description** field, type the message.
   b. To insert the MIB variables in the field, click **Open MIB Tree** . Set the message to include either the current or previous SNMP value, or the SNMP index, and click **OK**.
10. Click **Save**, then click **OK**.

Related concepts:

"Multibyte data in poll definitions" on page 10
If you are running Network Manager in a domain that uses multibyte characters such as Simplified Chinese, then you must ensure that Network Manager is configured to handle multibyte characters before you configure basic or generic threshold poll definitions.

"Poll policy scope" on page 2
The poll policy scope defines the devices or device interfaces to be polled.

Related reference:

Appendix A, “Default poll policies,” on page 173
Network Manager IP Edition provides a set of default poll policies. Use this information to familiarize yourself with these policies.

“Example generic threshold expression” on page 44
Use this example generic threshold expression to understand how to compose complex generic threshold expressions.

Appendix D, “Syntax for poll definition expressions,” on page 189
Use this information to understand how to build complex threshold expressions to use in basic and generic threshold poll definitions.

Changing chassis and interface ping poll definitions

Use the Poll Definition Editor to change chassis and interface ping poll definition types.

You can change some of the general properties of the poll definition and the properties that are associated with the poll definition type. However, you cannot change the poll definition type.

To change a chassis or interface ping poll definition:

1. Click **Administration** > **Network** > **Network Polling**.
2. Click the required poll definition. The poll definition must have the poll definition type Chassis Ping or Interface Ping.
3. In the Poll Definition Editor, under the **General** tab, complete the **General Properties** fields as follows:

   Name Specify a unique name for the poll definition. Only alphanumeric characters, spaces and underscores are allowed.

   Type This field is disabled. The Polling engine, ncp_poller, automatically populates this field once this poll definition is included as part of an enabled poll policy. The value assigned to the event ID is POLL-pollde

   Event ID
   This field is disabled. The Polling engine, ncp_poller, automatically populates this field once this poll definition is included as part of an enabled policy. The **Event ID** field is populated as follows:

   - If this is a new poll definition, then the **Event ID** field is populated with the value `POLL-`*`polldef`*, where *`polldef`* is the name of the current poll definition.
   - If you created a poll definition by copying an existing poll definition, then the **Event ID** contains the same value as the copied poll definition.

   Note: Some of the older default polls have **Event ID** fields that do not use the `POLL-`*`polldef`* naming convention.

   Event Severity
   Specify a valid number for the severity. The severity level must correspond to a valid severity level as defined in IBM Tivoli Netcool/OMNIbus. For a listing of available severity levels, refer to the *IBM Tivoli Network Manager IP Edition Network Troubleshooting Guide*

   Description
   Type a short description of the poll definition.
4. Click the **Classes** tab. In the **Classes** tree, select the check boxes of the required classes.

Attention: If you leave all classes unchecked, then the system polls all devices that match the scope defined in the poll policy that uses this poll definition.

5. Optional: Click the **Interface Filter** tab and build the filter against the required fields. The **Table** field is prepopulated with the `interfaces` table.
6. Click the **Ping** tab and complete the **Ping Properties** fields as follows:

   Timeout
   Specify, in milliseconds, how long the polling process should wait for a response from the target device before sending a new ping packet.

   Retries
   Specify how many times the polling process should attempt to ping the target device before giving up. When **Packet Loss** metric collection is enabled, the polling process sends this number of ping packets regardless of whether a response is received.

   Collect Ping Metrics

   Response Time
   Check the box to collect the time taken by devices to respond to a ping request.

   Packet Loss
   Check the box to collect data about lost packets.

   Payload Size
   Select the size of the ICMP packet to be used for the ping request. Select the default (32 bytes) or choose a custom size. This setting overrides the value of IcmpData in the `NcPollerSchema.cfg` configuration file.

   CAUTION:
   Using a size smaller than 32 bytes may result in packets being dropped.
7. Click **Save**, then click **OK**.

Related concepts:

"Poll policy scope" on page 2
The poll policy scope defines the devices or device interfaces to be polled.

Changing remote ping and link state poll definitions

Use the Poll Definition Editor to change the following poll definition types: Cisco remote ping, Juniper remote ping, and SNMP link state.

You can change some of the general properties of the poll definition and the properties that are associated with the poll definition type. However, you cannot change the poll definition type.

To change a remote ping poll definition or an SNMP link state poll definition:

1. Click **Administration** > **Network** > **Network Polling**.
2. Click the required poll definition. The poll definition must have one of the following poll definition types:
   - Cisco remote ping
   - Juniper remote ping
   - SNMP link state
3. In the Poll Definition Editor, under the **General** tab, complete the **General Properties** fields as follows:

Name Specify a unique name for the poll definition. Only alphanumeric characters, spaces and underscores are allowed.

Type This field is disabled. The Polling engine, ncp_poller, automatically populates this field once this poll definition is included as part of an enabled poll policy. The value assigned to the event ID is POLL-pollde

Event ID
This field is disabled. The Polling engine, ncp_poller, automatically populates this field once this poll definition is included as part of an enabled policy. The **Event ID** field is populated as follows:

- If this is a new poll definition, then the **Event ID** field is populated with the value `POLL-polldef`, where *polldef* is the name of the current poll definition.
- If you created a poll definition by copying an existing poll definition, then the **Event ID** contains the same value as the copied poll definition.

Note: Some of the older default polls have **Event ID** fields that do not use the `POLL-polldef` naming convention.

Event Severity
Specify a valid number for the severity. The severity level must correspond to a valid severity level as defined in IBM Tivoli Netcool/OMNIbus. For a listing of available severity levels, refer to the *IBM Tivoli Network Manager IP Edition Network Troubleshooting Guide*

Description
Type a short description of the poll definition.

4. Click the **Classes** tab. In the **Classes** tree, select the check boxes of the required classes.

   Attention: If you leave all classes unchecked, then the system polls all devices that match the scope defined in the poll policy that uses this poll definition.
5. Optional: Click the **Interface Filter** tab and build the filter against the required fields. The **Table** field is prepopulated with the `interfaces` table.
6. Click **Save**, then click **OK**.

Related concepts:

"Poll policy scope" on page 2
The poll policy scope defines the devices or device interfaces to be polled.

Related tasks:

"Configuring Link State polling" on page 65
Specify how the **ncp_poller** process determines the initial state for Link State polls when there is no existing event. The **ncp_poller** process can use the first poll to determine the initial state of Link State polls, or assume a Clear state.

Related reference:

Appendix A, "Default poll policies," on page 173
Network Manager IP Edition provides a set of default poll policies. Use this information to familiarize yourself with these policies.

Example customized poll definition

Use this example of a customized poll definition to help you copy an existing definition and customize it to your requirements.

Scenario

You require a poll definition that alerts the operator when a device is using too much of its processing power. You want an event to be generated if the average CPU usage exceeds 80%, and the event to be cleared if the average CPU usage falls below 80%.

Required settings

To create a poll definition that responds to the requirements described in "Scenario," make the following settings:

- On the **New Poll Definition Type Selection** panel, select `Basic Threshold`.
- On the **Poll Data** tab of the Poll Definition Editor make the following settings:
  - Ensure that **Single OID** is selected.
  - Complete the **Single OID** area as shown the following example. The bold text denotes entries that you must type; normal text denotes selections that you make from lists:

    ```
    current   avgBusy5
    ```

- On the **Threshold** tab of the Poll Definition Editor make the following settings:
  - Complete the **Trigger Threshold** area as shown the following example. The bold text denotes entries that you must type; normal text denotes selections that you make from lists. The italic text denotes items that you cannot edit:

    ```
    current   avgBusy5   >=   80
    ```

  - Complete the **Description** field as follows:
    1. Type `CPU usage high (avgBusy5=)`.
    2. Position the cursor inside the closing parenthesis and click **Add MIB Object** .
    3. Navigate the following path: `iso/org/dod/internet/private/enterprises/cisco/local/lsystem/avgBusy5`.
    4. Select **Current SNMP Value** and click **Insert**.
  - Complete the **Clear Threshold** area as shown the following example. The bold text denotes entries that you must type; normal text denotes selections that you make from lists. The italic text denotes items that you cannot edit:

    ```
    current   avgBusy5   <   80
    ```

  - Complete the **Description** field as follows:
    1. Type `CPU usage high (avgBusy5=)`.
    2. Position the cursor inside the closing parenthesis and click **Add MIB Object** .
    3. Navigate the following path: `iso/org/dod/internet/private/enterprises/cisco/local/lsystem/avgBusy5`.
    4. Select **Current SNMP Value** and click **Insert**.

Related concepts:

"Multibyte data in poll definitions" on page 10
If you are running Network Manager in a domain that uses multibyte characters such as Simplified Chinese, then you must ensure that Network Manager is configured to handle multibyte characters before you configure basic or generic threshold poll definitions.

Example basic threshold expression

Use this example basic threshold expression to understand how to compose complex basic threshold expressions.

Sample: snmpInBandwidth

The snmpInBandwidth poll definition is one of the default poll definitions. This poll definition defines the checking of incoming bandwidth utilization. An alert is raised when incoming bandwidth usage exceeds 40%. The following expression shows how to use `eval` statements to define this condition and is defined within the **Poll Data** tab under the **Expression** radio button.

```
((eval(long64,"&SNMP.DELTA.ifInOctets") / eval(long64,"&POLL.POLLINTERVAL"))
/(eval(long64,"&SNMP.VALUE.ifSpeed")))
*800
```

Within the **Threshold** tab, the threshold is set to trigger if the value of this expression is greater than 40.

Related concepts:

"Multibyte data in poll definitions" on page 10
If you are running Network Manager in a domain that uses multibyte characters such as Simplified Chinese, then you must ensure that Network Manager is configured to handle multibyte characters before you configure basic or generic threshold poll definitions.

Related tasks:

"Changing basic threshold poll definitions" on page 36
Use the Poll Definition Editor to change basic threshold poll definitions.

"Creating basic threshold poll definitions" on page 23
Create a basic threshold poll definition to run simple formulas against MIB variables, or to create threshold polls with interface-level filtering.

Related reference:

Appendix D, "Syntax for poll definition expressions," on page 189
Use this information to understand how to build complex threshold expressions to use in basic and generic threshold poll definitions.

Example generic threshold expression

Use this example generic threshold expression to understand how to compose complex generic threshold expressions.

Sample: memoryPoll

The memoryPoll poll definition is one of the default poll definitions. This poll definition defines the checking of memory-pool usage for Cisco devices. An alert is raised when the memory pool usage exceeds 80%. The following expression shows how to use `eval` statements to define this condition and is defined within the **Trigger Threshold** tab under the **Advanced** radio button.

```
((eval(int,"&SNMP.VALUE.ciscoMemoryPoolValid") = 1)
AND
((eval(long64,"&SNMP.VALUE.ciscoMemoryPoolUsed")/
(eval(long64,"&SNMP.VALUE.ciscoMemoryPoolFree") +
eval(long64,"&SNMP.VALUE.ciscoMemoryPoolUsed")))*100
> 80))
```

Related concepts:

"Multibyte data in poll definitions" on page 10
If you are running Network Manager in a domain that uses multibyte characters such as Simplified Chinese, then you must ensure that Network Manager is configured to handle multibyte characters before you configure basic or generic threshold poll definitions.

Related tasks:

"Changing generic threshold poll definitions" on page 38
Use the Poll Definition Editor to change generic threshold poll definitions.

"Creating generic threshold poll definitions" on page 25
Use the Poll Definition Editor to create new generic threshold poll definitions.

Related reference:

Appendix D, "Syntax for poll definition expressions," on page 189
Use this information to understand how to build complex threshold expressions to use in basic and generic threshold poll definitions.

Chapter 6. Deleting poll policies

Delete poll policies when they are no longer required.

To delete a poll policy:

1. Click **Administration** > **Network** > **Network Polling**.
2. In the **Select** column, select the required poll policies.
3. Click **Delete** .
4. Click **OK** to confirm the deletion.

The selected poll policies are deleted.

© Copyright IBM Corp. 2006, 2016

Chapter 7. Deleting poll definitions

Delete poll definitions when they are no longer required.

To delete a poll definition:

1. Click **Administration** > **Network** > **Network Polling**.
2. In the **Select** column, select the required poll definitions.
3. Click **Delete** .
4. Click **OK** to confirm the deletion.

The selected poll definitions are deleted.

© Copyright IBM Corp. 2006, 2016

Chapter 8. Managing adaptive polling

Adaptive polls dynamically react to events on the network. You can create adaptive polls that manage a wide range of network problem scenarios.

Related concepts:

"Adaptive polling plug-in" on page 119
Use this information to understand plug-in prerequisites, how the adaptive polling plug-in populates fields in the activeEvent table, as well as configuration details associated with the plug-in. The activeEvent table is in the NCMONITOR schema.

Related tasks:

"Configuring how many events the poller reads" on page 63
Large increases in the number of network events can slow down the polling process, **ncp_poller**. For adaptive polling, you can minimize the performance impact of sudden increases in events by defining how many events the poller reads.

Related reference:

"Plugin descriptions" on page 118
Use this information to understand what each Event Gateway plugin does.

Adaptive polling scenarios

Network Manager provides default adaptive polls to automatically handle key network problem scenarios. Use these default adaptive polls to understand how to create your own adaptive polls.

Rapid confirmation that device is really down

Use the adaptive polling described in this scenario to determine as quickly as possible when a device is really down. You can activate this adaptive polling by enabling the ConfirmDeviceDown poll policy.

Rationale

The standard Default Chassis Ping poll policy polls all chassis devices in the current network domain every two minutes. For various reasons healthy devices sometimes fail to respond to this poll policy, generating misleading NmosPingFail events in the **Active Event List (AEL)**. These events are not cleared until a successful ping response at least two minutes later. During that time these misleading events might cause network operators to take unnecessary action.

The adaptive polling described in this scenario avoids the risk of unnecessary network operations activity by accelerating pinging of all devices on which an NmosPingFail event is first raised. These devices are pinged by a separate poll policy every 10 seconds for three minutes, with the intention of clearing the event as soon as the device responds. Those devices that still exhibit an NmosPingFail event after three minutes of accelerated pinging are considered to be really down, and are no longer pinged. The network operator can take action on these devices, or automations can be written to take relevant action, with the confidence that the events are actionable.

Note: The three minutes are enforced by specifying as policy scope devices with an NmosPingFail event and a Tally value of less than 18. The Tally value is

© Copyright IBM Corp. 2006, 2016

calculated by assuming that a faulty device will fail to respond to all accelerated polls. Each minute there are six accelerated ping polls, so in a three minute period there will be 18, and the Tally value of those devices that are still not responding will reach 18.

The accelerated pinging values are fully configurable. For example, you can specify a 20 second accelerated ping polling interval (instead of a 10 second interval) to lighten the load on the network. You could also continue accelerated polling for longer than three minutes (by increasing the Tally value) if you require a longer time to verify that devices are really down.

Chained poll policies

The adaptive poll described in this scenario is made up of the following chained poll policies:

Default Chassis Ping
: This poll policy pings all devices in the network every two minutes. It is enabled by default.

ConfirmDeviceDown
: This poll policy provides accelerated pinging at 10 second intervals of devices that fail to respond to the Default Chassis Ping poll policy. The ConfirmDeviceDown policy must be assigned to a network view.

  Important: The ConfirmDeviceDown poll policy is disabled by default. You must enable this poll policy in order to activate the adaptive polling described in this scenario.

Scenario walkthrough

The following section walks you through this adaptive poll.

1. The Default Chassis Ping poll policy polls all chassis device in the current network domain. You can modify the scope of this policy by modifying the associated network views and device filter.
2. An NmosPingFail event is generated for all devices that do not respond to the Default Chassis Ping policy.
3. Devices that have an associated NmosPingFail event are automatically added to the Initial Ping Fail Events network view. The Initial Ping Fail Events network view is a Filtered network view that is defined as follows:

   ```
   EventId = NmosPingFail
   Tally < 18
   ```

   The Initial Ping Fail Events network view can be found in the Network Views, in the Network View tree, in the **Monitoring Views** > **Initial Ping Fail Events** node. For more information on the nodes in the network view tree, see the *IBM Tivoli Network Manager IP Edition Network Troubleshooting Guide*.

   You can change the duration of accelerated polling by modifying the `Tally < 18` filter clause. For example, if you want to increase the duration of accelerated polling to five minutes, then change this clause to `Tally < 30`. This value is determined by the following calculation based on a 10 second polling interval: each minute there are six accelerated ping polls, so in a five minute period there will be 30. For more information on changing event-filtered network views, see the *IBM Tivoli Network Manager IP Edition Network Visualization Setup Guide*.

4. The ConfirmDeviceDown poll policy polls all devices within the Initial Ping Fail Events network view every 10 seconds. Each time the device does not respond, another NmosPingFail is received for the device, and the NmosPingFail Tally value for the device is incremented.

   Important: By default you have to take the following actions in order to activate the adaptive polling described in this scenario:

   - The ConfirmDeviceDown poll policy is disabled by default. You must enable this poll policy.
   - The ConfirmDeviceDown poll policy has no scope by default. You must assign the Initial Ping Fail Events network view as the scope of the ConfirmDeviceDown poll policy.

   You can change the frequency of accelerated ping polling by editing the ConfirmDeviceDown poll policy and modifying the interval in seconds between poll operations. For example, if you want to decrease the polling frequency to 20 second polling intervals (three polls per minute rather than six), then open the ConfirmDeviceDown poll policy in the Poll Policy Editor and set the **Poll Interval** value to 20.

   Note: Changing the poll policy interval changes the Tally value for the same accelerated polling duration. For example, if you change the **Poll Interval** value to 20 this decreases the frequency of accelerated ping polling to 3 polls per minute. In this case the associated Tally value for a three minutes of accelerated polling is 9. This value is determined by the following calculation based on a 20 second polling interval: each minute there are three accelerated ping polls, so in a three minute period there will be 9.

5. Accelerated polling of devices can conclude in one of the following ways:

   Healthy device
   During the three-minute accelerated polling period a successful ping response is received for a device. The NmosPingFail event for that device is cleared and will subsequently be deleted from the **Active Event List (AEL)**. The device is automatically removed from the Initial Ping Fail Events network view, and therefore is no longer subject to accelerated polling.

   Faulty device
   The device continues to be ping polled until the end of the three-minute period. The Tally value for the event on that device reaches 18 and the device is automatically removed from the Initial Ping Fail Events network view, and therefore is no longer subject to accelerated polling. The **Active Event List (AEL)** now contains an actionable event, that is, an NmosPingFail event with a tally of 18 or greater.

Related tasks:

Chapter 2, “Enabling and disabling polls,” on page 11
To activate Network Manager polling, you must enable the poll policies. If a network entity is off the network, disable the poll policy that polls that entity.

“Changing poll policies” on page 31
Use the Poll Policy Editor to change the settings of existing poll policies.

“Creating adaptive polls” on page 56
Create adaptive polls to enable the system to dynamically react to events on the network.

Related reference:

Appendix A, “Default poll policies,” on page 173
Network Manager IP Edition provides a set of default poll policies. Use this information to familiarize yourself with these policies.

Appendix B, “Default poll definitions,” on page 179
Network Manager IP Edition provides a number of default poll definitions that fulfil the most common polling requirements.

Rapid confirmation of a threshold violation

Use the adaptive polling defined in this scenario to use slow SNMP polling most of the time to create minimum impact on your network devices, and accelerate polling when a threshold has actually been breached. You can activate this adaptive polling by enabling the ConfirmHighDiscardRate poll policy.

Rationale

The standard HighDiscardRate poll policy performs an SNMP threshold poll on all routers in the current network domain every 30 minutes to determine if the percentage packet discard rate on any of the router interfaces exceeds 5%. If the poll policy detects that any one interface on a router has exceeded the threshold, then the poll generates a POLL-HighDiscardRate event for the router. Healthy router interfaces might occasionally be busy and need to drop packets, causing them to breach the 5% threshold during any one 30 minute interval, generating misleading HighDiscardRate events in the **Active Event List (AEL)**. These events are not cleared until a successful POLL-HighDiscardRate poll policy response occurs at least thirty minutes later. During that time these misleading events might cause network operators to take unnecessary action.

The adaptive polling described in this scenario avoids the risk of unnecessary network operations activity by accelerating SNMP polling of all devices on which a POLL-HighDiscardRate event is first raised. These devices are polled by a separate SNMP poll policy every five minutes, with the intention of clearing the event as soon as all interfaces on the device respond with a percentage packet discard rate that falls within the 5% threshold. Those devices that continue to exhibit the POLL-HighDiscardRate event are confirmed as having one or more faulty interfaces. The network operator can take action on these devices, or automations can be written to take relevant action, with the confidence that the events are actionable.

The accelerated polling values are fully configurable. For example, you can specify a 10 minute accelerated SNMP polling interval (instead of a 5 minute interval) to lighten the load on the network.

Chained poll policies

The adaptive poll described in this scenario is made up of the following chained poll policies:

HighDiscardRate
: This poll policy determines whether an interface on a device is dropping more than a minimum percentage of the total packets that it is processing. The policy polls for this information every 30 minutes.

ConfirmHighDiscardRate
: This poll policy provides accelerated SNMP polling at five minutes

intervals of devices that have at least one interface that breached the 5% packet discard rate threshold. The ConfirmHighDiscardRate policy scope must be assigned to a network view.

Important: The ConfirmHighDiscardRate poll policy is disabled by default. You must enable this poll policy in order to activate the adaptive polling described in this scenario.

Scenario walkthrough

The following section walks you through this adaptive poll.

1. The HighDiscardRate SNMP poll policy performs an SNMP threshold poll on all routers in the current network domain every 30 minutes. You can modify the scope of this policy by modifying the associated network views and device filter.
2. A POLL-HighDiscardRate event is generated for all devices that have at least one interface that breached the 5% packet discard rate threshold.
3. Devices that have an associated POLL-HighDiscardRate event are automatically added to the Devices that have at least one interface event for HighDiscardRate network view. This network view is a Filtered network view that is defined as follows:

   ```
   EventId = POLL-HighDiscardRate
   ```

   The Devices that have at least one interface event for HighDiscardRate network view can be found in the Network Views, in the Network View tree, in the **Monitoring Views** > **Devices that have at least one interface event for HighDiscardRate** node. For more information on the nodes in the network view tree, see the *IBM Tivoli Network Manager IP Edition Network Troubleshooting Guide*.
4. The ConfirmHighDiscardRate poll policy polls all devices within the Devices that have at least one interface event for HighDiscardRate network view every five minutes.

   Important: By default you have to take the following actions in order to activate the adaptive polling described in this scenario:

   - The ConfirmHighDiscardRate poll policy is disabled by default. You must enable this poll policy.
   - The ConfirmHighDiscardRate poll policy has no scope by default. You must assign the Devices that have at least one interface event for HighDiscardRate network view as the scope of the ConfirmHighDiscardRate poll policy.

   Important: The ConfirmHighDiscardRate poll policy is disabled by default. You must enable this poll policy in order to activate the adaptive polling described in this scenario.

   You can change the frequency of accelerated SNMP threshold polling by editing the ConfirmHighDiscardRate poll policy and modifying the interval in seconds between poll operations. For example, if you want to decrease the polling frequency to 10 minute polling intervals, then open the ConfirmHighDiscardRate poll policy in the Poll Policy Editor and set the **Poll Interval** value to 600.
5. Accelerated SNMP threshold polling of devices can conclude in one of the following ways:

Healthy device

All interfaces on the device respond to the accelerated poll with a percentage packet discard rate that falls within the 5% threshold. The POLL-HighDiscardRate event for that device is cleared and will subsequently be deleted from the **Active Event List (AEL)**. The device is automatically removed from the Devices that have at least one interface event for HighDiscardRate network view, and therefore is no longer subject to accelerated polling.

Faulty device

At least one interface on the device continues to respond to the accelerated SNMP poll with a breach of the 5% packet discard rate. The POLL-HighDiscardRate event remains in the **Active Event List (AEL)** with a continually increasing Tally value. The **Active Event List (AEL)** now contains an actionable POLL-HighDiscardRate event.

Related tasks:

Chapter 2, "Enabling and disabling polls," on page 11
To activate Network Manager polling, you must enable the poll policies. If a network entity is off the network, disable the poll policy that polls that entity.

"Changing poll policies" on page 31
Use the Poll Policy Editor to change the settings of existing poll policies.

"Creating adaptive polls"
Create adaptive polls to enable the system to dynamically react to events on the network.

Related reference:

Appendix A, "Default poll policies," on page 173
Network Manager IP Edition provides a set of default poll policies. Use this information to familiarize yourself with these policies.

Appendix B, "Default poll definitions," on page 179
Network Manager IP Edition provides a number of default poll definitions that fulfil the most common polling requirements.

Creating adaptive polls

Create adaptive polls to enable the system to dynamically react to events on the network.

Before creating an adaptive poll, you must first identify a network condition that would benefit from adaptive polling. For example, the adaptive polling provided by default with Network Manager provides accelerated polling of a selected set of devices in order to do one of the following:

- Confirm that a device that failed an ICMP ping is really down.
- Confirm that a threshold on a device has really been violated.

Proceed as follows to create an adaptive poll which creates a chain of two poll policies. The Confirm device down and Confirm threshold violation columns provide examples from the adaptive polling provided by default with Network Manager.

Procedure	Confirm device down	Confirm threshold violation
1. Identify an existing poll policy that retrieves an error condition on devices in your network.	Poll policy used: **Default Chassis Ping** Performs ping polling on all devices in the network domain every two minutes.	Poll policy used: **HighDiscardRate** Determines whether an interface on a device is dropping more than a minimum percentage of the total packets that it is processing. Polls for this information every 30 minutes.
2. Create an event-filtered network view that filters devices based on the event generated by the poll that you identified in the previous step. The devices in this network view usually have an associated error condition that can be further diagnosed by more intense polling. For information on creating event-filtered network views, see the *IBM Tivoli Network Manager IP Edition Network Visualization Setup Guide*.	Network view: **Monitoring Views** > **Initial Ping Fail Events** Contains all devices on which an NmosPingFail event has been raised and the Tally value is less than a specified value. The NmosPingFail event is raised on events that fail the Default Chassis Ping poll policy.	Network view: **Monitoring Views** > **Devices that have at least one interface event for HighDiscardRate** Determines whether an interface on a device is dropping more than a minimum percentage of the total packets that it is processing.
3. Create a poll policy that has as its scope the network view that you created in the previous step and that provides a more intense polling of the devices in that network view. The aim of more intensely polling these devices is to diagnose the problem further as a prelude to further action.	Poll policy: **ConfirmDeviceDown** Purpose of intense polling: accelerate ping polling of devices in the Initial Ping Fail Events network view to 10 second poll intervals in order to identify the devices that are really down. Healthy devices provide a successful response to this intense polling and their events are cleared.	Poll policy: **ConfirmHighDiscardRate** Purpose of intense polling: Accelerate polling of devices in the Devices that have at least one interface event for HighDiscardRate network view in order to provide more timely information before reacting. This poll policy continues to generate the POLL_HighDiscardRate events, thus confirming the problem on a device, or issues a resolved event that clears the error event and thus removes the associated device from the HighDiscardRate view.

In Step 2, in the *Confirm device down* column, the *Initial Ping Fail Events* network view includes an exit criterion implemented using the Tally value: once the Tally value for an event goes beyond a specified value, the related device is automatically removed from the network view. This is useful when you want to accelerate polling for a limited time period in order to establish a condition on that device. Once the condition is established the device can be removed from the view. For example, in the case of the default settings for this network view, you want to

accelerate polling on devices that have failed ping polling for three minutes only. If the device still has an associated NmosPingFail event after three minutes, then the device is confirmed as being down.

It is also possible to chain more than two poll policies by creating extra network views and poll policies and chaining them as appropriate to respond to network conditions and to perform the required diagnosis.

Related concepts:

"Rapid confirmation that device is really down" on page 51
Use the adaptive polling described in this scenario to determine as quickly as possible when a device is really down. You can activate this adaptive polling by enabling the ConfirmDeviceDown poll policy.

"Rapid confirmation of a threshold violation" on page 54
Use the adaptive polling defined in this scenario to use slow SNMP polling most of the time to create minimum impact on your network devices, and accelerate polling when a threshold has actually been breached. You can activate this adaptive polling by enabling the ConfirmHighDiscardRate poll policy.

Related tasks:

Chapter 3, "Creating polls," on page 13
Create polls if the existing default poll policies and definitions do not meet your requirements. Either customize a copy of an existing or default poll, or create a new poll from scratch.

Chapter 9. Administering network polling

Use the command-line interface to perform a wide range of polling administration tasks, including managing the multiple poller feature, copying network polls across network domains, suspending network polling, enabling and disabling polls, retrieving poll status, and refreshing polls.

You can also configure the SNMP Helper to use the GetBulk operation when SNMP v2 or v3 is used. Use of the GetBulk operation improves polling efficiency. For more information, see the *IBM Tivoli Network Manager IP Edition Installation and Configuration Guide*.

Administering polls

Use the command-line interface to administer poll policies.

Speeding up ncp_poller startup by not checking SNMP credentials

Fix Pack 3

Disabling the testing of SNMP access credentials on startup of **ncp_poller** can improve startup times.

If you are sure that your access credentials for the devices you want to poll are accurate, you can configure the polling process, **ncp_poller**, not to check them on startup.

1. Back up and edit the file `NCHOME/etc/precision/NcPollerSchema.DOMAIN.cfg`
2. Locate the line that defines the value of the DiscoverInitialAccess option.
3. If you want the **ncp_poller** process to check the SNMP credentials each time it starts up, leave the value of DiscoverInitialAccess set to 1.
4. To turn off the initial checking of SNMP access credentials, set the value of DiscoverInitialAccess set to `0`. The **ncp_poller** process still tests SNMP credentials for a given device if there is a poll failure.
5. Restart the **ncp_poller** process.

Retrieving poll status

Use the **itnm_poller.pl** script to display the status of poll policies.

The domain provided on the command-line interface (CLI) must have an entry in the domainMgr NCIM table.

For more information on the **itnm_poller.pl** script, see the *IBM Tivoli Network Manager IP Edition Administration Guide*.

1. Change to the `$NCHOME/precision/scripts/perl/scripts` directory and locate the **itnm_poller.pl** script.
2. Run the **itnm_poller.pl** script program to retrieve the status of poll policies. The following example shows how to retrieve the status of all poll policies.

   ```
   ncp_perl itnm_poller.pl -domain NCOMS -status all
   ```

© Copyright IBM Corp. 2006, 2016

You can also retrieve the status of only realtime or static poll polices by specifying `-status realtime` or `-status static`.

Enabling and disabling polls

Use the itnm_poller.pl script to enable and disable individual poll policies.

The domain provided on the command-line interface (CLI) must have an entry in the domainMgr NCIM table.

For more information on the itnm_poller.pl script, see the *IBM Tivoli Network Manager IP Edition Administration Guide*.

1. Change to the `$NCHOME/precision/scripts/perl/scripts` directory and locate the itnm_poller.pl script.
2. Run the itnm_poller.pl script program to retrieve the status of the various poll policies, as shown in the following example.

   ```
   ncp_perl itnm_poller.pl -domain NCOMS -status all
   ```

   In the output of the command you can see the ID for each poll policy. Make a note of the ID for the poll policy that you want to enable or disable.
3. Run the itnm_poller.pl script program to enable or disable individual poll policies, specifying the poll policy ID as a parameter. The following example shows how to enable a poll policy with a policy ID of 10.

   ```
   ncp_perl itnm_poller.pl -domain NCOMS -enable 10
   ```

   The following example shows how to disable a poll policy with a policy ID of 15.

   ```
   ncp_perl itnm_poller.pl -domain NCOMS -disable 15
   ```

Refreshing polls

Use the itnm_poller.pl script to refresh a poll policy configuration and its entity list.

The domain provided on the command-line interface (CLI) must have an entry in the NCIM topology database domainMgr table.

For more information on the itnm_poller.pl script, see the *IBM Tivoli Network Manager IP Edition Administration Guide*.

1. Change to the `$NCHOME/precision/scripts/perl/scripts` directory and locate the itnm_poller.pl script.
2. Run the itnm_poller.pl script program to retrieve the status of the various poll policies, as shown in the following example.

   ```
   ncp_perl itnm_poller.pl -domain NCOMS -status all
   ```

   In the output of the command you can see the ID for each poll policy. Make a note of the ID for the poll policy that you want to enable or disable.
3. Run the itnm_poller.pl script program to refresh a single poll policy, or of poll policies. The following example shows how to refresh a poll policy with a policy ID of 10.

   ```
   ncp_perl itnm_poller.pl -domain NCOMS -refresh 10
   ```

   The following example shows how to refresh all poll policies.

   ```
   ncp_perl itnm_poller.pl -domain NCOMS -refresh all
   ```

Copying polls across domains

Use the `get_policies.pl` program to copy your poll policies from one network domain to another, or between a file and a domain.

If you provide a domain name with either the `-to` or `-from` options, you must be connected to the NCIM and NCMONITOR databases. The connections are created based on the settings in the `DbLogins.DOMAIN.cfg` file, or the `DbLogins.cfg` file if no domain-specific file is found.

The domain provided on the command-line interface (CLI) must have an entry in the domainMgr NCIM table.

1. Change to the `NCHOME/precision/scripts/perl/scripts` directory and locate the `get_policies.pl` program.
2. Run the `get_policies.pl` program to copy poll policies. The following table describes the possible actions and what to enter on the CLI.

Action	Entry on the CLI
Copy all poll policies directly from one domain to another	`ncp_perl get_policies.pl -from domain=SOURCE -to domain=DESTINATION -ncim_password NCIM_password -ncmonitor_password NCMONITOR_password`
Copy only selected poll policies from one domain to another	`ncp_perl get_policies.pl -from domain=SOURCE -to domain=DESTINATION -policy "policy_name1" -policy "policy_name2"`
Copy all the poll policies from a domain to an XML file	`ncp_perl get_policies.pl -from domain=DOMAIN_name -to file=filename.xml -password NCIM_password`
Copy all the poll policies from an XML file to a domain	`ncp_perl get_policies.pl -from file=filename.xml -to domain=DOMAIN_name`
Copy only selected poll policies from a domain to a file	`ncp_perl get_policies.pl -from domain=DOMAIN_name -to file=filename.xml -policy "policy_name1" -policy "policy_name2"`

Polling suspension options

Set a device or component to an unmanaged state to suspend it from Network Manager network polling.

When a device or component is unmanaged, it is not polled by Network Manager, and events are not generated for the device or component. Also, no root cause analysis (RCA) is performed on events for that device.

Restrictions

Setting the device to "unmanaged" affects only Network Manager polling, it does not stop other polls from retrieving information from the device or its components and raising alerts where applicable. However, the alerts from other sources are tagged as unmanaged to indicate that the device or component they were raised against are in maintenance state.

For more information on unmanaged device settings, see the *IBM Tivoli Network Manager IP Edition Discovery Guide*.

Options to suspend polling

To suspend a device or its component from active poll operations, you have the following options:

Set a device or component to unmanaged
You have the following options:

- Use the Network Views or the Hop View
- Use the **Unmanage node** tool

If you set a device to unmanaged, then all components within that device also become unmanaged. If an individual component within a device is set to unmanaged, then only that component and any components contained within become unmanaged, the device itself remains in managed state. A component can be managed or unmanaged only if the device within which it resides is in managed state. Also, setting the management interface associated with the chassis to unmanaged state does not suspend polls for the whole device.

Set specific interface types to be permanently unmanaged
This means that the specified interface types are not polled by Network Manager. Also, you can set new devices not to be polled initially after being discovered and added to the topology. These settings are determined by the `TagManagedEntities.stch` file and the NCIM database tables.

For more information on the Network Views and the Hop View, see the *IBM Tivoli Network Manager IP Edition Network Visualization Setup Guide*. For more information on setting devices and components to an unmanaged state, see the *IBM Tivoli Network Manager IP Edition Network Troubleshooting Guide*.

Adjusting polling bandwidth

You can configure how much data is transferred by the Polling engine, ncp_poller, and how often. You might want to adjust polling bandwidth to avoid network congestion or to reduce the impact of large numbers of polling events occurring simultaneously.

Adjusting the size of poll data queues

Fix Pack 4

Database problems or heavy polling loads can cause pollers to run out of memory if the queue of poll data for writing to the NCPOLLDATA database becomes too large. To monitor and avoid this problem, you can set a limit on the size of the queue. The limit is defined as the number of batches. If the limit is exceeded, you are alerted in the log and in the Active Event List (AEL). Then, the poller reduces the queue size by discarding the queued data. The discarded data is written to another file so that you can retrieve it and write it to the database.

Set the poller debug level to 4.

1. To set the limit, in the `$NCHOME/etc/precision/NcPollerSchema.cfg` file for your poller, set the **PollDataQueueLimit** parameter to a suitable number of batches. The following example shows how to set the limit to 50 batches of queued data:

   ```
   update config.properties set PollDataQueueLimit = 50;
   ```

2. Monitor the poller log file in `$NCHOME/log/precision` and the AEL for messages and alerts.

When the queue exceeds the limit that is specified by the **PollDataQueueLimit** parameter, the following actions are taken:

- In the AEL, an alert is displayed. For example:

  ```
  ItnmPollerPolicyDataQueueFull: poller NCOMS; poll data queue has
  exceeded capacity, off loading data to file
  ```

- A message is written to the log. For example:

  ```
  2013-04-19T12:37:58 [CDataQueue::ProcessValue] Queue size exceeds threshold,
  discarding data: policyId:122:templateId:39:monitoredInstanceId:2212
  :monitoredObjectId:3:pollTime:1387232947:
  tdwTime:1131216222944000:errorCode:111:value:0
  ```

- The poll data is discarded from the queue and written to a `$NCHOME/log/precision/ncp_poller.`*poller name*`.`*domain*`.polldata` file, for example, `ncp_poller.`*poller name*`.`*domain*`.data`. The data is written as SQL INSERT statements. For example:

  ```
  INSERT INTO pollData( MONITOREDOBJECTID, MONITOREDINSTID, POLLTIME, TDWTIME,
  ERRORCODE, VALUE )  VALUES ( 3, 2828, 1387232686, 1131216172446000, 100, 714 );
  INSERT INTO pollData( MONITOREDOBJECTID, MONITOREDINSTID, POLLTIME, TDWTIME,
  ERRORCODE, VALUE )  VALUES ( 3, 2829, 1387232686, 1131216172446000, 100, 715 );
  ```

If the limit is exceeded:

- Resolve the size of the poll data queues. For example, create more pollers, or contact your database administrator.
- Clear the alert from the AEL.
- Write the data from the `.polldata` file to the NCPOLLDATA database. Then, clear the `.polldata` file. The file is not cleared or rotated automatically.

Related tasks:

"Setting up additional pollers" on page 66
Set up an additional poller on the Network Manager server if one poller is not sufficient to handle your network load. To add more pollers, register and create new instances of the **ncp_poller** process.

"Monitoring poller capacity" on page 74
To prevent problems from occurring on pollers, monitor the poller metrics by outputting them on the command-line interface. The metrics are displayed as bar charts. The metrics show when a poller is reaching capacity, for example, if poor database performance is causing it to fall behind.

Configuring how many events the poller reads

Fix Pack 4

Large increases in the number of network events can slow down the polling process, **ncp_poller**. For adaptive polling, you can minimize the performance impact of sudden increases in events by defining how many events the poller reads.

Adaptive **ncp_poller** processes, read events from the NCMONITOR.activeEvent table into memory at intervals to determine the initial state of polling events. You can put an upper limit on the number of events that the poller reads. Additional events are not read into memory. When the number of events decreases below the limit again, the poller resumes reading events.

1. Back up and edit the `$NCHOME/etc/precision/NcPollerSchema.cfg` file.
2. Edit, uncomment, or add the following line:

   ```
   update config.properties set EventThrottle = number_of_events
   ```

Where *number_of_events* is the maximum number of events that the poller reads. If this line is not present, there is no limit to the number of events that the poller attempts to process.

3. Save and close the file.

Related concepts:

"Adaptive polling plug-in" on page 119
Use this information to understand plug-in prerequisites, how the adaptive polling plug-in populates fields in the activeEvent table, as well as configuration details associated with the plug-in. The activeEvent table is in the NCMONITOR schema.

Related tasks:

Chapter 8, "Managing adaptive polling," on page 51
Adaptive polls dynamically react to events on the network. You can create adaptive polls that manage a wide range of network problem scenarios.

Related reference:

"Plugin descriptions" on page 118
Use this information to understand what each Event Gateway plugin does.

Changing the interval for checking poll policy membership size

You can change the interval for checking poll policy membership size for all poll policies.

Before changing the interval for checking poll policy membership size, you should consider the following:

- A longer interval means less load on the network due to longer time between checks.
- A shorter interval means that ncp_poller is kept more up to date with changes, especially network view membership size.

1. Edit the following configuration file: `$NCHOME/etc/precision/NcPollerSchema.cfg`.
2. Add the following line to the end of the file:

   ```
   update config.properties set PolicyUpdateInterval= update_interval;
   ```

   Where: *update_interval* is the interval for checking poll policy membership size, in seconds. The default value is 30 seconds.
3. Restart the Polling engine, ncp_poller.

Enabling and disabling network view updates for poll policies

By default network view updates are enabled for all poll policies. You can disable network view updates if you are polling stable network views. This will avoid any performance issues associated with the update.

1. Edit the following configuration file: `$NCHOME/etc/precision/NcPollerSchema.cfg`.
2. Add the following line to the end of the file:

   ```
   update config.properties set UpdateNetworkViewCache = enable_or_disable_update;
   ```

   Where: *enable_or_disable_update* is either `1` (enable) or `0` (disable). The default value is `1` (enable).
3. Restart the Polling engine, ncp_poller.

Configuring Link State polling

Specify how the **ncp_poller** process determines the initial state for Link State polls when there is no existing event. The **ncp_poller** process can use the first poll to determine the initial state of Link State polls, or assume a Clear state.

1. Edit the `NCHOME/etc/precision/NcPollerSchema.cfg` file.
2. Locate the `config.properties` section.
3. Edit the value for **UseFirstPollForInitialState**.
   - Set to `0` to set the initial state to Clear.
   - Set to `1` to configure the initial state to be determined by the first poll.

Related concepts:

"Link state polling" on page 5
Link state polling monitors changes to the status of the ifOperStatus and ifAdminStatus interface MIB variables. If the value of these variables changes between poll intervals, an event is raised.

Related tasks:

"Changing remote ping and link state poll definitions" on page 41
Use the Poll Definition Editor to change the following poll definition types: Cisco remote ping, Juniper remote ping, and SNMP link state.

Configuring SNMP threshold polls

Fix Pack 5

SNMP threshold polls that measure differences in counter variables, such as bandwidth variables, can record incorrect values if the monitored device restarts.

You can configure the polling process to discard any polls that are performed while a device is restarting.

To configure the polling process in this way, complete the following steps:

1. Back up and edit the following file on the server where the Network Manager core components are installed: `$NCHOME/etc/precision/NcPollerSchema.cfg`
2. Add the following line:

   ```
   update config.properties property set DropWrappedPolls = 1
   ```

3. Restart the **ncp_poller** process.

Administering multiple pollers

If multiple pollers are needed to poll your network, you can set up Network Manager to administer the multiple poller feature. You can add pollers or remove pollers, or use a poller ID to associate a specific poller with a policy.

Multiple poller overview

You can use multiple pollers to scale your polling to more devices and for performance reasons.

The poller process has two functions: to poll network devices, and to perform administrative tasks such as data pruning and updating caches. If a single poller is polling many devices on a regular basis, consider using multiple pollers for polling.

Note: If you have more than one poller and you want to configure them differently, you can create domain-specific and poller-specific versions of the poller configuration file, `NcPollerSchema.cfg`. To create a domain-specific version, add the domain name to the file name; for example `NcPollerSchema.`*`DOMAIN`*`.cfg`. To create a poller-specific version, add the name of the poller and the name of the domain to the file name; for example, `NcPollerSchema`*`POLLERNAME`*`.`*`DOMAIN`*`.cfg`. A poller instance uses the most specific configuration file available.

You can also avoid performance issues by using a separate poller to perform administrative functions. Start only one poller with the `-admin` option, and do not use this poller for polling network devices. Start the other pollers with the `-noadmin` option, and use these pollers for network polling. If you are using failover, replicate the same arrangement of pollers on the backup server.

Each poller is registered with a name that the administrator can use to associate a policy with a poller.

Restriction: Distributing polling over multiple servers is not supported. Start all pollers on the Network Manager server.

You can monitor the polling engines using the Network Manager Health views in IBM Tivoli Monitoring.

For additional information, see the *IBM Tivoli Monitoring for Tivoli Network Manager IP User's Guide*.

Monitor Policy editor

When multiple pollers are deployed, the administrator controls which poller is used by a policy. By default, each policy uses the default poller on the Network Manager server. The administrator can select a different poller from a list of registered pollers.

Real-time MIB graphing

The poller used for real-time polling is defined by the administrator.

Setting up additional pollers

Set up an additional poller on the Network Manager server if one poller is not sufficient to handle your network load. To add more pollers, register and create new instances of the **`ncp_poller`** process.

By default, a single poller is automatically installed during the installation of the product. This default poller has no name.

1. To register the poller, run the `$ITNMHOME/bin/ncp_poller` command. For example, to register a poller that is called POLL2344 on the NCOMS domain run the command as follows:

   ```
   ncp_poller -domain NCOMS -register -name POLL2344
   ```

2. In the `$NCHOME/etc/precision/CtrlServices.cfg` file, create the new poller instance.
   a. Make a copy of the entry for the **ncp_poller** process.
   b. In this copy, change the **serviceName** parameter to the name of the poller that you just created.
   c. Change the other **ncp_poller** options as required.

      Fix Pack 3 Ensure that one of the pollers is started with the `-admin` option, which configures the poller to perform essential administrative functions, and the other pollers are started with the `-noadmin` option.See "Samples" for example **ncp_poller** entries in the `CtrlServices.cfg` file.

3. Restart the **ncp_ctrl** process in the specified domain. All running **ncp_*** processes are restarted, including the new poller.

Samples

The following example creates a poller in the NCOMS domain that is called snmpPoller that you can use for SNMP queries:

```
insert into services.inTray
(
    serviceName,
    binaryName,
    servicePath,
    domainName,
    argList,
    dependsOn,
    retryCount
)
values
(
    "snmpPoller",
    "ncp_poller",
    "$PRECISION_HOME/platform/$PLATFORM/bin",
    "$PRECISION_DOMAIN",
    [ "-domain" , "$PRECISION_DOMAIN" , "-latency" , "60000", "-debug", "0",
"-messagelevel", "warn", "-name", "snmpPoller"],
    [ "nco_p_ncpmonitor", "ncp_g_event" ],
    5
);
```

The following example creates a poller that is called PingPoller that you can use for ping queries:

```
insert into services.inTray
(
    serviceName,
    binaryName,
    servicePath,
    domainName,
    argList,
    dependsOn,
    retryCount
)
values
(
    "PingPoller",
    "ncp_poller",
```

```
    "$PRECISION_HOME/platform/$PLATFORM/bin",
    "$PRECISION_DOMAIN",
    [ "-domain" , "$PRECISION_DOMAIN" , "-latency" , "60000", "-debug", "0",
"-messagelevel", "warn", "-name", "PingPoller"],
    [ "nco_p_ncpmonitor", "ncp_g_event" ],
    5
);
```

Fix Pack 3 The following example creates a poller in the NCOMS domain that you can use for ping queries. This poller is set as the admin poller.

```
insert into services.inTray
(
    serviceName,
    binaryName,
    servicePath,
    domainName,
    argList,
    dependsOn,
    retryCount
)
values
(
    "PingPoller",
    "ncp_poller",
    "$PRECISION_HOME/platform/$PLATFORM/bin",
    "$PRECISION_DOMAIN",
    [ "-domain" , "$PRECISION_DOMAIN" , "-latency" , "60000", "-debug", "0",
"-messagelevel", "warn", "-name", "PingPoller" "-admin"],
    [ "nco_p_ncpmonitor", "ncp_g_event" ],
    5
);
```

Fix Pack 3 The following example creates a poller in the NCOMS domain that you can use for SNMP queries. This poller runs in the same environment as the poller in the previous example and so is set to run as `-noadmin`.

```
insert into services.inTray
(
    serviceName,
    binaryName,
    servicePath,
    domainName,
    argList,
    dependsOn,
    retryCount
)
values
(
    "snmpPoller",
    "ncp_poller",
    "$PRECISION_HOME/platform/$PLATFORM/bin",
    "$PRECISION_DOMAIN",
    [ "-domain" , "$PRECISION_DOMAIN" , "-latency" , "60000", "-debug", "0",
"-messagelevel", "warn", "-name", "snmpPoller" "-noadmin"
 ],
    [ "nco_p_ncpmonitor", "ncp_g_event" ],
    5
);
```

After you add the pollers:

1. Restart the product.
2. Edit the poll definitions.
3. Assign polls by setting the **Assign to Poller Instance** field in the poll policies.

Related tasks:

"Changing poll definitions" on page 36
Change existing poll definitions to customize them for your polling requirements. You change poll definitions in the Poll Definition Editor; the steps you follow differ depending on the *poll definition type*.

"Changing poll policies" on page 31
Use the Poll Policy Editor to change the settings of existing poll policies.

Assigning a poller for MIB graphing

If you have defined additional pollers, and want to assign a poller to handle MIB graphing, edit the properties file

You must have created the poller that you want to use for MIB graphing before you edit the file.

To assign a poller for MIB graphing:

1. Open the following file: `ITNMHOME/profiles/TIPProfile/etc/tnm/tnm.properties`.
2. Change the value of the **`tnm.graph.poller`** parameter from `DEFAULT_POLLER` to the name of the required poller.
3. Save and close the file.

Removing a poller

If a poller is not being used to monitor the network, you can remove the poller from the monitoring system.

To remove a poller from the monitoring system, deregister the poller from the command line. The poller is removed from the Monitor Policy editor.

Complete the following tasks to remove a poller from the monitoring system.

1. Before removing the poller, edit each active policy associated with the poller that you want to remove. From the Monitor Policy Editor, edit the policies to use another valid poller.
2. Stop all running Network Manager processes.
3. From the Network Manager server, run the following command.

   `ncp_poller -deregister -domain [`*`domain_name`*`] -name [`*`poller_name`*`]`

   If there are no active policies associated with the poller, the poller is deregistered. If there are still active policies associated with the poller, the script does not deregister the poller.
4. If there are still active policies associated with the poller, either assign the policies to another poller, or run the script again using the `- force` option. If you run the script using the `- force` option, any poll policies associated with the poller are also deleted.
5. Edit the `CtrlServices.cfg` file to remove or uncomment the **`ncp_poller`** entry for the poller that you removed.
6. Restart the Network Manager processes.

Administering historical polled data

Use the command-line interface to administer historical polled data.

Storage capacity considerations

Use this information to understand the issues, guidelines, and calculation method associated with storage capacity. Understanding your storage capacity requirements can help you determine if Network Manager can accommodate an increase in the storage limit for historical polled data.

It is important to consider the amount of data you need to store as well as the rate at which it is collected. By default, the Network Manager poller maintains a pruning policy to maintain the latest 5 million database rows. This value can be increased to 20 million or more database rows, but any increase beyond 20 million will impact performance for reporting and may require tuning and maintenance attention to the database.

The term "sustained rate of data storage" describes the average speed with which the data can be inserted into the database and a daily rate will be used for the purposes of the calculations in the example shown below. While faster rates may be possible, it is important to allow room for the inevitable brief interruptions along the pipeline such as network interruptions or database slowdowns due to peak loads, restarts, or interruptions.

The sustained rate of data storage depends on a number of factors, including:

- Number of polled entities
- Number of metrics polled
- Frequency of polling
- Number of policies
- Number of pollers
- Number of database rows
- Database performance (for large rates of data storage, a slow database will have an impact)

Insertion rates to the local NCPOLLDATA historical polled data database (hereinafter referred to as the NCPOLLDATA database) above 7 million database rows per day are typically not satisfactory over time. A single polling policy will store about 1 million database rows per day, and you can use multiple policies and multiple pollers within the suggested overall range. You can use Network Manager to increase the number of pollers. To avoid excessive impact on the NCPOLLDATA database, follow these guidelines:

- Use no more than 3 pollers for storing data.
- Create one ITM agent instance per Network Manager poller.

Throughput to the Tivoli Data Warehouse is up to a total of 7 million database rows per day with all pollers. Exceeding this limit shortens the tolerance limit for outages and might cause loss of data. While rates higher than this can be achieved, you need to allow for possible error conditions on the network link and transfer processes. Network Manager will tolerate and recover from transfer outages without loss of data for a short period of time. This time period depends on a combination of factors such as data rate, disk space, and length of time associated with the transfer outage. The longer the transfer outage, the longer it takes for the system to recover.

Storage capacity guidelines for historical polled data

Use this information to understand the guidelines associated with calculating the storage capacity to help you determine if Network Manager can accommodate an increase in the storage limit for historical polled data.

To calculate the rate of database rows to be stored in the NCPOLLDATA historical polled data database or the Tivoli Data Warehouse, follow these guidelines:

- For historical polled data associated with devices (SNMP or ICMP poll):

  ```
  Number of device level database rows per day =
  number_of_devices * 60 * 24/polling_freq
  ```

  where:

  – *number_of_devices* — Specifies the number of devices for which you want to perform an SNMP or ICMP poll.

  – `60 * 24` — Specifies the number of minutes in a day.

  – *polling_freq* — Specifies, in minutes, the polling frequency for the historical polled data associated with the devices to be polled. Examples of historical polled data associated with devices in an SNMP poll include memory utilization and CPU cycles.

    For an ICMP poll, you can specify these historical polled data items: pings for up/down status, response time, and packet loss.

- For historical polled data associated with network interfaces (SNMP or ICMP poll):

  ```
  Number of network interface level database rows per day =
  number_of_network_interfaces * number_of_data_points *
  60 * 24/polling_freq
  ```

  where:

  – *number_of_network_interfaces* — Specifies the number of network interfaces for which you want to perform an SNMP or ICMP poll.

  – *number_of_data_points* — Specifies the number of data points. Assign a data point for each historical polled data associated with the network interfaces for which you want to perform an SNMP poll. For example, if you want historical polled data for bandwidth and `ifInDiscards` then you would assign the value `2` to *number_of_data_points*.

    For an ICMP poll, you can specify these historical polled data items: pings for up/down status, response time, and packet loss. Count one data point if you select the pings for up/down status historical polled data; count two data points if you select the pings for up/down status and response time historical polled data; select three data points if you select all three of these historical polled data. For example, if you want historical polled data for pings for up/down status and response time then you would assign the value `2` to *number_of_data_points*.

  – `60 * 24` — Specifies the number of minutes in a day.

  – *polling_freq* — Specifies, in minutes, the polling frequency for the historical polled data associated with the network interfaces to be polled. Examples of historical polled data associated with network interfaces include bandwidth and MIB objects, such as, `ifInDiscards` and `ifInErrors` from the SNMP MIB II interfaces table, and so on.

    For an ICMP poll, you can specify these historical polled data items: pings for up/down status, response time, and packet loss.

Example of storage capacity calculation

Use this information to learn how to calculate the storage capacity to help you determine if Network Manager can accommodate an increase in the storage limit for historical polled data.

A user wants to poll 1000 devices with an average of 5 network interfaces for the following historical polled data with the following polling intervals:

Device level

Historical polled data: memory utilization, 5 minute intervals

Interface level

Historical polled data and polling frequency (SNMP): `ifInDiscards`, 10 minute intervals

Historical polled data and polling frequency (SNMP): `ifInErrors`, 10 minute intervals

Historical polled data and polling frequency (SNMP): bandwidth, 10 minute intervals

Historical polled data and polling frequency (ICMP): Pings for up/down status and response time, 5 minute intervals

Based on these device level and interface level polling requirements, a user would calculate the daily rate of database rows using the previously described guidelines.

Note: In the example, the SNMP poll specifies a count of three data points for the `ifInDiscards`, `ifInErrors`, and bandwidth historical polled data. The ICMP poll specifies a count of two data points for the up/down status and response time historical polled data.

```
Number of device level database rows per day (SNMP) =
1000 devices * 60 * 24/5 polls per day = 288,000

Number of network interface level database rows per day (SNMP) =
1000 * 5 interfaces * 3 data points * 60 * 24/10 polls per day = 2,160,000

Number of ICMP database rows per day =
1000 devices * 5 interfaces * 2 data points * 60 * 24/5 polls per day  = 2,880,000

Total database rows per day = 5,328,000
```

The previous example shows a total database rows per day of 5,328,000, which is within the upper guidance of 7 million database rows per day. Thus, this example shows a scenario that results in maintaining about 4 days of raw data after increasing the storage limit for historical polled data to 21 or 22 million database rows.

Increasing the storage limit for historical polled data

You can increase the storage limit for historical polled data. This enables Performance reports to display a greater amount of historical polled data.

The Network Manager Performance reports are operationally focused and designed for short term diagnostics. Thus, features typical for a full performance management product (for example, Tivoli Netcool Performance Manager) such as optimized data storage and routine data gathering over extended periods that are required for capacity planning and regulatory reporting are not a goal with the Network Manager historical reports. You can store up to 20 million database rows in either the local database or Tivoli Data Warehouse. Depending on your

hardware and database performance, you might see degradation in the storage and reporting performance above 20 million database rows. You may also need the services of your Database Administrator to optimize and run statistics and transaction log maintenance regularly on the database, with increasing storage rates and table size.

Note: By default, Network Manager implements the NCPOLLDATA database using a database schema within the NCIM topology database.

The storage limit for historical polled data is set by default to a conservative value of 5 million database rows. You can increase this value; however, this can lead to a degradation in the performance of the Performance reports.

Data pruning occurs in batches so that rows are not constantly being removed from the database. When the number of rows exceeds the limit by 10,000 then rows are removed until the number of rows reaches the limit.

To increase the storage limit for historical polled data, complete the following steps:

1. Locate the following file:
   - UNIX `$NCHOME/etc/precision/NcPollerSchema.cfg`
   - Windows `%NCHOME%\etc\precision\NcPollerSchema.cfg`
2. Locate the following line:
   ```
   insert into config.pruning (MAXPOLLDATAROWS) values (5000000);
   ```
3. Change the value of `MAXPOLLDATAROWS`.

   Attention: Increasing this value might lead to a degradation in report performance.

Deleting historical polled data

If there is too much historical polled data in the database, you can use the poll data pruning script to delete a subset of the historical polled data.

You can delete a subset of the historical polled data that is defined by criteria such as number of rows, or the age of the data.

The poll data pruning script is installed with the IBM Tivoli Monitoring for IBM Tivoli Network Manager IP Edition.

To run the poll data pruning script, complete the following tasks.

1. On the server where IBM Tivoli Monitoring for IBM Tivoli Network Manager IP Edition is installed, change to the `NCHOME/precision/products/tnm/bin` directory.
2. Start the script using a command line similar to the following:
   - UNIX itnm_polldata_pruning [-help] -username *user* -password *password* {-days *days* [-domain *domain*] [-policy *policy*]} | {-rows *rows*} [-prompt]
   - Windows itnm_polldata_pruning.bat [-help] -username *user* -password *password* {-days *days* [-domain *domain*] [-policy *policy*]} | {-rows *rows*} [-prompt

   The command line parameters are listed in the following table.

Table 6. Command line parameters for the itnm_polldata_pruning script

Parameter	Description
-days *days*	The number of days to keep historical polled data in the database. To delete historical polled data, you must use either the -days or the -rows parameters.
-domain *domain*	Optional. Deletes rows that have the given domain. Used with -days.
-help	Shows the help for the script.
-password *password*	The password of a user with access to the database that stores the historical polled data.
-policy *policy*	Optional. Deletes rows that have the given policy name. Used with -days.
-prompt	Optional. Prompts the user before deleting any historical polled data.
-rows *rows*	The number of rows to delete, starting from the oldest data. Cannot be used with the -days parameter.

Monitoring poller capacity

Fix Pack 4

To prevent problems from occurring on pollers, monitor the poller metrics by outputting them on the command-line interface. The metrics are displayed as bar charts. The metrics show when a poller is reaching capacity, for example, if poor database performance is causing it to fall behind.

By default, the metrics are written to a trace in `NCHOME/log/precision` every 2 minutes. There is one trace for each poller. The file name of the trace has the format `ncp_poller.SnmpPoller.`*`domain`*`.metrics` for the default poller and `ncp_poller.SnmpPoller.`*`pollername`*`.`*`domain`*`.metrics` for all other pollers. For example, `ncp_poller.SnmpPoller.Poller23507.NCOMS.metrics`. The following table describes the metrics.

Table 7. Poller metrics

Metric	Measures	Measured in (the units of the y-axis of each bar chart)
Health	The percentage of devices that are polled during a policy cycle. If this value is 100%, the poller is working properly. If the value is below 100%, not all the devices are polled during the polling interval. The poller cannot keep up with policy load.	%

Table 7. Poller metrics (continued)

Metric	Measures	Measured in (the units of the y-axis of each bar chart)
Memory	The memory that the poller is using. Memory usage increases as more devices are discovered or more policies are enabled.	MB
BatchQueueSize	The number of batches that are waiting for a thread.	Count
PollDataQueueSize	The number of INSERT statements that are queued to the NCPOLLDATA database. Shows whether the poller is successfully storing polling data.	Count
PollDataRowCount	The number of rows in the ncpolldata.polldata table after the poller has pruned the data. This table stores historical poll data. Only the default poller prunes the data. This metric is useful only if historical polling is used.	Count

Do not run the script with the `-metrics` option and the `-status` option simultaneously.

Ensure that the terminal on which you output the bar charts has a minimum width of 140 characters. Otherwise, the bar chart does not display properly because of the line wrapping.

Run the `itnm_poller.pl` script as shown in the examples. The following example shows how to display the charts for the default poller, on the NCOMS domain, from the most recent time stamp, over the default 4 hour period:

```
ncp_perl itnm_poller.pl -domain NCOMS -metrics
```

The following example shows how to run the script for the same domain for a specific poller, over the last 12 hours:

```
ncp_perl itnm_poller.pl -domain NCOMS -poller Poller23507 -metrics -window 12
```

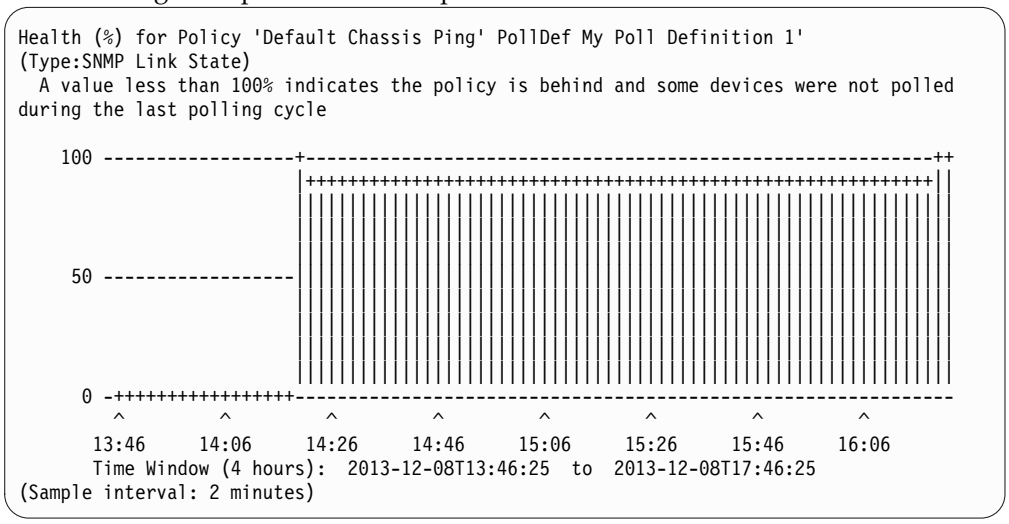
The following example shows how to run the script for the same domain and poller, from a specific time stamp, over a period of 8 hours:

```
ncp_perl itnm_poller.pl -domain NCOMS -poller Poller23507 -metrics
-timestamp 2013-12-10T17:30:36 -window 8
```

Pay close attention to the scale of the y-axis. A bar chart that appears flat for a long period, for example 24 hours, might show differences in values when the chart is viewed over a shorter period, for example 4 hours.

Example

The following example shows a sample chart for the Health metric.



Review the problems and possible causes in the following table and take action as appropriate.

Table 8. Possible causes of problems with poller metrics

Metric	Problem	Possible cause	Actions
Health	Value is consistently below 100%.	The percentage can fall temporarily below 100% after the poller is started, or if change information is received from the MODEL database.	• Increase the polling interval by changing the poll policy • Add more pollers.
Memory	Memory grows unbounded	The connection to the database was lost. Alternatively, the polling load is too great to sustain, or the rate of data storage is too great to sustain.	• Contact your database administrator. • Add more pollers.

Table 8. Possible causes of problems with poller metrics (continued)

Metric	Problem	Possible cause	Actions
BatchQueue	The number of batches that are waiting for a thread is greater than 0 and increasing.	The number of threads is exhausted, which can indicate that the downstream SNMP dispatcher is close to capacity.	Although it is possible to increase the number of threads by setting the **BatchExtraThreads** property in the `NcPollerSchema.cfg` file, it is not the best solution. It is possible that increasing the number of threads worsens the problem. Safer solutions are as follows: • Add more pollers. • Contact your system administrator to investigate adding RAM to the host. **Tip:** Set a threshold on the number of batches that are in the queue for processing. You are alerted in the poller log when the threshold is breached.[1]
PollDataQueueSize	The number of INSERT statements in the queue grows exponentially.	The connection to the database was lost or the frequency of INSERT statements is greater than the poller can handle.	• Contact your database administrator. • Add more pollers.
PollDataRowCount	The number or rows exceeds the threshold after pruning is completed. The default threshold is 5,000,000 and the default pruning interval is 1 hour.	The polling load is too heavy and so the number of rows is too great to be pruned within the pruning interval. Alternatively, problems occurred in the database, which is causing problems with pruning.	Contact your database administrator.

Table 8. Possible causes of problems with poller metrics (continued)

Metric	Problem	Possible cause	Actions

Table notes:
1. To set a threshold, change the value of the set **BatchQueueThreshold** property in the $NCHOME/etc/precision/NcPollerSchema.cfg to a suitable value. For example, to set the threshold to 10 batches of queued polls:
`update config.properties set BatchQueueThreshold = 10;`
When the queue exceeds the specified threshold, a message is written that is similar to the following example:
`2013-04-19T12:37:58:Poller:NCOMS:DataQueueSize:10;`			

If an error is displayed, check in the $NCHOME/etc/precision/NcPollerSchema.cfg file whether the **CollectPollerMetrics** parameter is disabled. This parameter is enabled by default, but, if it is disabled, enable it. You can use the OQL interface to enable the parameter at run time. For example:

```
ncp_oql -domain NCOMS -service SnmpPoller -poller Poller23507
-query "update config.properties set CollectPollerMetrics=1;"
```

Related tasks:

"Setting up additional pollers" on page 66
Set up an additional poller on the Network Manager server if one poller is not sufficient to handle your network load. To add more pollers, register and create new instances of the **ncp_poller** process.

"Changing poll policies" on page 31
Use the Poll Policy Editor to change the settings of existing poll policies.

"Adjusting the size of poll data queues" on page 62
Database problems or heavy polling loads can cause pollers to run out of memory if the queue of poll data for writing to the NCPOLLDATA database becomes too large. To monitor and avoid this problem, you can set a limit on the size of the queue. The limit is defined as the number of batches. If the limit is exceeded, you are alerted in the log and in the Active Event List (AEL). Then, the poller reduces the queue size by discarding the queued data. The discarded data is written to another file so that you can retrieve it and write it to the database.

Querying the status of entities

To troubleshoot pollers, query the monitors.current OQL table. This table contains records of the entities that are in the scope of poll policies. Use the **ncp_oql** command to query the table. The records include information such as the last polling time of entities. For unpolled entities, the output includes the managed status or shows that the entity is considered a duplicate IP address.

Important: Fix Pack 4 By default queries on the monitors.current table output chassis records but omit interface records for monitored entities. To output both chassis and interface records, apply fix pack 4.

- To query the monitors.current table, run the **ncp_oql** command as shown in the following example for the MASTER domain.

  ```
  ncp_oql -domain MASTER -service SnmpPoller -tabular
  -query "select * from monitors.current;"
  ```

  The chassis records are output, as shown in the following example.

```
| ENTITYID | ENTITYNAME | ACCESSIPADDRESS | MAINNODEENTITYID | MAINNODEENTITYNAME

| MAINNODEADDRESS | ENTITYMANAGED | CHASSISMANAGED | ISMANAGED | LASTPOLLTIME

| LASTPOLLINTERVAL | LASTPOLLFAILURE | TEMPLATEID | TEMPLATENAME | POLICYID

| POLICYNAME | CURRENTTIME |
| 27 | 172.30.120.15 | 172.30.120.15 | 27 | 172.30.120.15 | 172.30.120.15 |

| | true | 2013-10-23 16:38:24 | 180 | 2013-10-23 16:38:24 | 47 | aNoSuchName

| 179 | aNoSuchName | 2013-10-23 16:39:06 |

( 1 record(s) : Transaction complete )
```

Fix Pack 4 The chassis records are output first, then the interface records, as shown in the following example.

```
| ENTITYID | ENTITYNAME | ACCESSIPADDRESS | MAINNODEENTITYID | MAINNODEENTITYNAME

| MAINNODEADDRESS | ENTITYMANAGED | CH ASSISMANAGED | ISMANAGED | LASTPOLLTIME

| LASTPOLLINTERVAL | LASTPOLLFAILURE | TEMPLATEID | TEMPLATENAME | POLICYID

| POLICYNAME | CURRENTTIME | ENTITYCLASS | LINGERTIME | ENTITYTYPE | IFINDEX

| IFTYPESTRING | IFNAME | IFDESCR | IFALIAS  | INSTANCESTR | MONITOREDINSTID

|
| 27 | 172.30.120.15

| 172.30.120.15 | 27 | 172.30.120.15 | 172.30.120.15 | |   | true

| 2013-10-23 16:38:24 | 180 | 2013-10-23 16:38:24 | 47 | aNoSuchName | 179

| a NoSuchName | 2013-10-23 16:40:47 | NULL | NULL | NULL | NULL | NULL | NULL

| NULL | NULL   | NULL | NULL | | 1714 | 172.30.120.15[ 19 ] | | 27

| 172.30.120.15 | 172.30.120.15 | |   | true | NULL | NULL | NULL | 47

| aNoSuchName | 179 | a NoSuchName | NULL | CiscoCat19xx | 3 | 2 | 19

| ethernet-csmacd | 19 | 19 |   | 19 | 386 | | 1726 | 172.30.120.15[ CPU ]

| 172.30.120.15 | 27 | 172.30.120.15 | 172.30.120.15 | |   | true | NULL

| NULL | NULL | 47 | aNoSuchName | 179 | a NoSuchName | NULL | CiscoCat19xx

| 3 | 2 | 37 | ethernet-csmacd | CPU | CPU |   | 37 | 1536 |

( 3 record(s) : Transaction complete )
```

- Fix Pack 4 To query the table for only chassis records, run the command as shown in the following example.

```
ncp_oql -domain MASTER -service SnmpPoller -tabular
-query "select * from monitors.current where ENTITYTYPE=1;
```

Chapter 10. Troubleshooting ping polling

Use this information to help you ensure that the important IP addresses in your network are being ping polled as expected by Network Manager and, if not, to provide information to resolve the problem.

By default, the tables and views defined in the `$NCHOME/precision/scripts/SQL/createPollLogTables.sql` file are added to the NCMONITOR schema as part of the Network Manager installation. These tables and views store the results of the diagnostic operations performed in this procedure.

Note: This mechanism is for ping polling policies only, not for SNMP polling policies.

This procedure includes a step for taking a snapshot of the current ping polling status within a specified network domainAfter starting the Network Manager polling process, you must allow at least two polling intervals before taking the first snapshot. A restart of the system is not necessary.

Given a set of IP addresses, the scripts described in this task provide insight into whether the poller is pinging those IP addresses, and if not, provide an indication of what the problem is. There are a number of reasons why IP addresses might not be being polled including the following:

- The device specified in the poll policy was not discovered.
- The device or interface is not included in any of the ping policy scopes.
- The device or interface has been marked as unmanaged from one of the topology visualization GUIs.
- The interface was marked as unmanaged at discovery time or determined to be unroutable.

For more information on the scripts described in this task, see the *IBM Tivoli Network Manager IP Edition Administration Guide*.

Note: These scripts are a verification and diagnostic tool only and have no effect on the actual polling of the devices; polling of devices is governed solely by the poll policies set up using the Network Polling GUI.

1. Add the list of IP addresses whose polling you want to monitor using the `ncp_upload_expected_ips.pl` script. Issue the following command: `$NCHOME/precision/bin/ncp_perl $NCHOME/precision/scripts/perl/scripts/ncp_upload_expected_ips.pl -domain DOMAIN_NAME -file FILENAME -password PASSWORD` Where:
   - *DOMAIN_NAME* is the name of domain that contains the IP addresses. The list of IP addresses will only be compared with IP addresses in this domain.
   - *FILENAME* is a plain text file of IP addresses, separated by whitespace (for example, one IP address per line). This only accepts IPv4 addresses. The file is expected to contain just IP addresses in standard dot notation.
   - *PASSWORD* is the database password used to access the NCIM and NCMONITOR schemas.

© Copyright IBM Corp. 2006, 2016

Note: Only repeat this operation if there are changes to the list of IP addresses whose polling you want to monitor. Otherwise, this is a one-time operation only per domain.

2. Periodically, or when required for troubleshooting purposes, take a snapshot of the current ping polling status within the domain specified in the previous step. Issue the following command:`$NCHOME/precision/bin/ncp_perl $NCHOME/precision/scripts/perl/scripts/ncp_ping_poller_snapshot.pl -domain DOMAIN_NAME -password PASSWORD` Where:
   - *DOMAIN_NAME* is the name of domain within which to take a snapshot of the current ping polling status.
   - *PASSWORD* is the database password used to access the NCIM and NCMONITOR schemas.

   The results of the operation are stored in the pollLog database table within the NCMONITOR schema.
3. Report on the entities that are not being polled.Run the report of the status of entities polled or not polled by the Network Manager polling process. Issue the following command: `$NCHOME/precision/bin/ncp_perl $NCHOME/precision/scripts/perl/scripts/ ncp_polling_exceptions.pl -domain DOMAIN_NAME [ -notpolled ] [-format LIST | REPORT ]` Where:
   - *DOMAIN_NAME* is the name of domain within which to report on current ping polling status.
   - `-notpolled`: this optional parameter outputs a list of IP addresses that are not polled as compared with the list of expected IP addresses. This output is in LIST format only.
   - `-format LIST | REPORT`: this optional parameter specifies whether the output should be in report or list format.

   This command outputs two lists: a list of IP addresses that you want to poll and a list of IP addresses that are not being polled. You can see at a glance if any of the IP addresses that you want to poll are not being polled.

Related reference:

"Ping polling status tables" on page 216
The NCMONITOR ping polling status tables enable diagnostic operations to be performed on network ping polling.

Chapter 11. About event enrichment and correlation

Network Manager uses the Event Gateway to correlate events with topology data and enrich events with a default set of topology fields. Once an event has been enriched, it is passed to plugin processes such as root-cause analysis (RCA) and failover, which take further action based on the data in the enriched event. The enriched event is also passed back to the ObjectServer.

Event enrichment

The event enrichment process occurs within the Event Gateway and is made up of distinct steps. Each of these steps can be customized.

Related concepts:

Appendix F, "Network Manager event categories," on page 207
The events that are raised by Network Manager fall into two categories: events about the network being monitored and events about Network Manager processes.

Quick reference for event enrichment

Use this information to understand how an event is processed as it passes through the Event Gateway.

Note: When accessing a Tivoli Netcool/OMNIbus ObjectServer that is protected by a firewall, you must specify an IDUC port and provide access to that port using the firewall.For more information on specifying an ObjectServer IDUC port, see the *IBM Tivoli Netcool/OMNIbus Administration Guide*.

The steps are described in the following table.

Table 9. Quick reference for event enrichment

Action	Further information	Data passed to next step
1. An event is received from the ObjectServer and the incoming event filter is applied to the event. The default filter checks the LocalNodeAlias field of the event. If the LocalNodeAlias field is not empty, then the event passes the filter and moves to Step 3. **Note:** The LocalNodeAlias field usually contains data that points to the main node device on which the event occurred. The precise data held by the LocalNodeAlias field varies, and can include the following: • IP address • DNS name • sysName	"Incoming event filter" on page 86	Event
2. The Event Gateway assigns a state to the event based on the Severity and Tally fields in the event. This event state is an internal Event Gateway representation and is used later by the plug-ins as part of the event subscription mechanism.	"Event states" on page 93	Event Event state

© Copyright IBM Corp. 2006, 2016

Table 9. Quick reference for event enrichment (continued)

Action	Further information	Data passed to next step
3. The incoming field filter is applied to the event. This field filter filters out alerts.status fields that do not participate in the Event Gateway processing.	"Incoming field filter" on page 89	Event with filtered fields Event state
4. The Event Gateway determines how to handle this event, by determining which event map to use. Event maps define how to handle an event. At the same time a numerical precedence value is associated with an event. This precedence value is used by the RCA plugin in cases where there are multiple events on the same entity. The event with the highest precedence value on the entity is used to suppress other events.	"Event map selection" on page 97 "Precedence value" on page 131	Event with filtered fields Event state Event map fields, such as event map name and event enrichment stitcher Precedence value
5. The Event Gateway determines the entity ID of the Network Manager server or of the ingress interface, the interface within the discovery scope from which network packets are transmitted to and from the Network Manager server. This value is used by the RCA plug-in to perform isolated suppression.	"Poller entity" on page 133	Event with filtered fields Event state Event map fields, such as event map name and event enrichment stitcher Precedence value PollerEntityId
6. The Event Gateway performs a topology lookup to retrieve entity data associated with this event, and then enriches the event using some of this entity data. To perform the topology lookup and event enrichment, the Event Gateway calls the stitcher defined in the event map.	"Event Gateway stitchers" on page 102	To Steps 8 and Step 9 Event with filtered fields and enriched fields Event state Precedence value PollerEntityId Return value from stitcher
7. The outgoing field filter is applied to the event. This filter only passes the fields enriched by the Event Gateway, and in particular, by the GwEnrichEvent stitcher rule. The enriched fields are placed on the Event Gateway queue, from where the data is sent to the ObjectServer at a configurable interval (default is 5 seconds).	"Outgoing field filter" on page 90 "Outgoing Event Gateway queue" on page 91	Fields enriched by Event Gateway

Table 9. Quick reference for event enrichment (continued)

Action	Further information	Data passed to next step
8. Based on the return value from the stitcher defined in the event map (Step 7), the Event Gateway determines whether to send the enriched event to the plug-ins. • If the return value is 1, then the Event Gateway sends the enriched event to the plug-ins. Go to the next step. • If the return value is 0, then the Event Gateway does not send the enriched event to the plug-ins. The event enrichment process for this event ends here.	"Event enrichment stitchers" on page 112	Event with filtered fields and enriched fields Event state Event map name Precedence value PollerEntityId
9. Each plug-in determines whether it is interested in the enriched event. It does this based on the event map name and the event state. The plug-ins that are interested in the event perform further event enrichment or take other action.	"Plugin descriptions" on page 118 "Plug-in subscriptions" on page 128	Event with filtered fields and enriched fields
10. On completion of processing, the enriched fields are placed on the Event Gateway queue, from where the data is sent to the ObjectServer at a configurable interval (default is 5 seconds).	"Outgoing Event Gateway queue" on page 91	To ObjectServer Fields enriched by plug-ins

Related concepts:

"Example: Default enrichment of a Tivoli Netcool/OMNIbus trap event" on page 115
Use this information to understand how a Tivoli Netcool/OMNIbus event is processed as it passes through the Event Gateway.

Related tasks:

"Modifying event map subscriptions" on page 163
You can change the event maps that a plug-in subscribes to. For example, if you add a new event map and want the system to perform RCA on events handled by that event map, then you must add that event map to the subscription list for the RCA plug-in.

Event filtering

Events are filtered at the beginning of the enrichment process when the events are received from the ObjectServer. Events are also filtered at the end of the process when the enriched events are sent back to the ObjectServer.

Incoming filter

The incoming filter ensures that only events and fields of interest are passed to the Event Gateway for event enrichment.

Incoming event filter:

The incoming event filter filters out events from the ObjectServer and only passes events that meet certain criteria.

The incoming event filter is used in both installations without failover, and in installations with failover. In the case of an installation with failover, the filter mechanism applies to events on the active domain, that is, the primary domain when the primary server is healthy, or the backup domain when the primary server is down.

The incoming event filter only passes events that meet the following conditions:

- The LocalNodeAlias field of the event is populated. This field should hold the IP address or DNS name of the related device.
- The domain name specified in the NmosDomainName field of the event is the same as the domain handled by the current Event Gateway. Alternatively, there is no domain associated with this event, and the NmosDomainName field is empty.

The incoming event filter can handle events in both single domain and multidomain systems.

Single domain and multidomain handling

In a single domain system, there is only one Event Gateway process. All events that have a domain set in the NmosDomainName field have the same domain assignment as this Event Gateway.

In a multiple domain system, there are multiple Event Gateway processes, one for each domain. Each Event Gateway process receives events from the ObjectServer and filters the events so that it only receives events for its own domain. The ObjectServer performs deduplication and matching of problem and resolution events based on the following alerts.status fields: AlertKey, Identifier, and Domain. The inclusion of the Domain field ensures that all deduplication and matching of problem and resolution events is domain specific.

Events with no domain

Events that come from sources outside of Network Manager, for example, Tivoli Netcool/OMNIbus syslog or trap probes, do not have a domain setting the first time that they pass through an Event Gateway. In a single domain system events with no domain setting pass the incoming event filter and the Event Gateway determines the domain associated with the event as part of the topology lookup performed during event enrichment.

In a multidomain system, the first Event Gateway that encounters the event with no domain passes it through the incoming event filter and proceeds to process it. However, if the topology lookup fails to find an entity for the event, then the event will be rejected by the Event Gateway without any event enrichment. The event is then picked up by a different Event Gateway, which also attempts to match the event to a device in its domain. This process continues until the event eventually is processed by an Event Gateway that is able to match an entity to the event.

Related reference:

"config.nco2ncp table" on page 234
The config.nco2ncp table is used to filter events being passed from Tivoli

Netcool/OMNIbus to Network Manager.

Incoming event filter: default configuration:

This example shows how the incoming event filter is configured in the EventGatewaySchema.cfg configuration file. This standard insert for the incoming event filter handles both single-domain and multi-domain systems.

The incoming event filter is configured in the EventGatewaySchema.cfg configuration file. This file is located at: `$NCHOME/etc/precision/EventGatewaySchema.cfg`.

The following table describes the relevant lines from this insert.

Table 10. Lines of code relevant to the incoming event filter

Line numbers	Description
1	Configure the incoming filter by making an insert into the config.nco2ncp table.
3	Specify an insert into the EventFilter field of the config.nco2ncp table.
9 - 10	Set the filter as follows: `"LocalNodeAlias <> '' and (NmosDomainName = '$DOMAIN_NAME' or NmosDomainName = '')"` This clause checks that the LocalNodeAlias column of the event has been populated. In addition, the filter checks whether the domain name specified in the event (held in the NmosDomainName field) is the same as the domain that is handled by this Event Gateway (held in the `$DOMAIN_NAME` variable). If there is a match, or if the event has no associated domain (`NmosDomainName - ''`), and the LocalNodeAlias column has been populated, then the event passes the filter.

The following table describes the relevant lines from this insert.

```
1]    insert into config.nco2ncp
2]    (
3]        EventFilter,
4]        StandbyEventFilter,
5]        FieldFilter
6]    )
7]    values
8]    (
9]        "LocalNodeAlias <> '' and (NmosDomainName =
10]            '$DOMAIN_NAME' or NmosDomainName = '')",
11]       "EventId in ('ItnmHealthChk', 'ItnmDatabaseConnection')",
12]       [
13]           "Acknowledged",
14]           "AlertGroup",
15]           "EventId",
16]           "FirstOccurrence",
17]           "LastOccurrence",
18]           "LocalNodeAlias",
19]           "LocalPriObj",
20]           "LocalRootObj",
21]           "Manager",
22]           "NmosCauseType",
23]           "NmosDomainName",
24]           "NmosEntityId",
25]           "NmosEventMap",
26]           "NmosManagedStatus",
27]           "NmosObjInst",
28]           "NmosSerial",
29]           "Node",
30]           "RemoteNodeAlias",
31]           "EventId",
32]           "Serial",
33]           "ServerName",
34]           "Severity",
35]           "Summary",
36]           "SuppressEscl",
37]           "Tally",
38]           "Type"
39]       ]
40]   );
```

Standby filter:

In a failover deployment, the standby filter is used by the backup domain in a failover pair. That means the backup domain when the primary is active, or the primary domain if the backup is active. The standby filter only allows health check (ItnmHealthCheck) events through the Event Gateway. These events are passed to the Failover plugin and tell the system to switch back to primary mode. Note that for failover behaviour, any modifications to this filter must still ensure that the standby filter accepts health check events.

When the primary server is active, the Event Gateway on the backup server does not perform any event enrichment. When the primary server is down and the backup server is active, the Event Gateway on the backup server performs event enrichment of events in the ObjectServer.

The standby filter is configured in the EventGatewaySchema.cfg configuration file. This file is located at: `$NCHOME/etc/precision/EventGatewaySchema.cfg`.

The example listed in “Incoming event filter: default configuration” on page 87 shows how the standby filter is configured in the EventGatewaySchema.cfg configuration file.

The section of code that is relevant to the standby filter is listed in the following lines. This insert configures the Event Gateway to only pass ItnmHealthCheck events when the primary server is down and the backup server is active.

The following table describes the relevant lines from this insert:

Table 11. Lines of code relevant to the standby filter

Line numbers	Description
1	Configure the incoming filter by making an insert into the config.nco2ncp table.
4	Specify an insert into the StandbyEventFilter field of the config.nco2ncp table.
11	Set the filter as follows: `"EventId in ('ItnmHealthChk', 'ItnmDatabaseConnection')",` This clause only passes events that have the event ID set to the value `ItnmHealthChk` or `ItnmDatabaseConnection`.

Related reference:

“config.nco2ncp table” on page 234
The config.nco2ncp table is used to filter events being passed from Tivoli Netcool/OMNIbus to Network Manager.

Incoming field filter:

For each event that passes the incoming event filter, the field filter specifies a subset of alerts.status fields that are passed through to the event enrichment process. If the field filter is empty then all alerts.status fields are are passed through to the event enrichment process. The purpose of this filter is to limit the fields passed through to the minimum required set in order to lighten the processing load.

The incoming event filter is configured in the EventGatewaySchema.cfg configuration file. This file is located at: `$NCHOME/etc/precision/EventGatewaySchema.cfg`.

The example listed in “Incoming event filter: default configuration” on page 87 shows how the incoming field filter is configured in the EventGatewaySchema.cfg configuration file.

The section of code that is relevant to the incoming field filter is listed in the following lines from the example. For each event that the ObjectServer passes to the Event Gateway, the incoming field filter specifies a subset of alerts.status fields that are passed through to the event enrichment process.

The following table describes the relevant lines from this example:

Table 12. Lines of code relevant to the incoming field filter

Line numbers	Description
1	Configure the incoming filter by making an insert into the config.ncp2nco table.
5	Specify an insert into the FieldFilter field of the config.ncp2nco table.
11-38	Only pass the alerts.status fields specified in lines 13 to 38. **Note:** If you configure extra event enrichment, you might need to add fields to this list.

Outgoing field filter

The outgoing field filter defines the set of ObjectServer fields that may be updated by the Event Gateway.

Event enrichment is performed by the `GwEnrichEvent()` stitcher rule. Of the fields enriched with that rule, only those listed in this filter are transmitted to the Object Server. Some event enrichment is intended purely to provide data for use by Event Gateway plug-ins. The enriched fields intended only for use by plug-ins do not need to be specified in the outgoing field filter. The outgoing field filter works on any data passed to the `GwEnrichEvent()` rule

Note: If you customize event enrichment to add extra enriched fields to the event, then you must update the outgoing field filter to include these extra enriched fields. For example, if you want enrich events so that any alert on a Cisco routers alerts is increased to critical severity (severity 5) then you must add Severity to the list of fields in the outgoing field filter.

The outgoing field filter is configured in the EventGatewaySchema.cfg configuration file. This file is located at: `$NCHOME/etc/precision/EventGatewaySchema.cfg`.

The following example shows how the outgoing field filter is configured in the EventGatewaySchema.cfg configuration file.

The section of code that is relevant to the outgoing field filter is listed in the following lines from the example. Each time the Event Gateway passes enriched fields back to the ObjectServer, only the fields listed in this filter are transmitted. Any other fields are ignored.

The following table describes the relevant lines from this example:

Table 13. Lines of code relevant to the incoming event filter

Line numbers	Description
1	Configure the outgoing field filter by making an insert into the config.ncp2nco table.
3	Specify an insert into the FieldFilter field of the config.ncp2nco table.
5-14	Only pass the fields specified in lines 5 to 14 back to the ObjectServer. **Note:** If you configure extra event enrichment, you must add fields to this list.

```
1]    insert into config.ncp2nco
2]    (
3]        FieldFilter
4]    )
5]    values
6]    (
7]        [
8]            "NmosCauseType",
9]            "NmosDomainName",
10]           "NmosEntityId",
11]           "NmosManagedStatus",
12]           "NmosObjInst",
13]           "NmosSerial"
14]       ]
15]   );
```

Related tasks:

"Example: Enriching an event with main node device location" on page 154
You can configure event enrichment so that the location of the main node device associated with an event is added to a field in the event.

"Example: Enriching an event with interface name" on page 155
You can configure event enrichment so that for all interface events, the name of the interface on which the event occurred is added to a field in the event.

Related reference:

"config.ncp2nco table" on page 236
The config.ncp2nco table is used to filter and map events passed from Network Manager IP Edition to Tivoli Netcool/OMNIbus.

Outgoing Event Gateway queue:

The outgoing Event Gateway queue receives enriched events from the Event Gateway stitchers (main event enrichment) and from the plug-ins. In order to minimize the number of updates and hence minimize the load on the ObjectServer, updates to the Object Server are placed in a queue, aggregated, and sent to the ObjectServer at a specified interval. The default is 5 seconds.

The outgoing Event Gateway queue receives enriched events from both the Event Gateway stitchers (main event enrichment) and from the plug-ins, as shown in the following diagram.

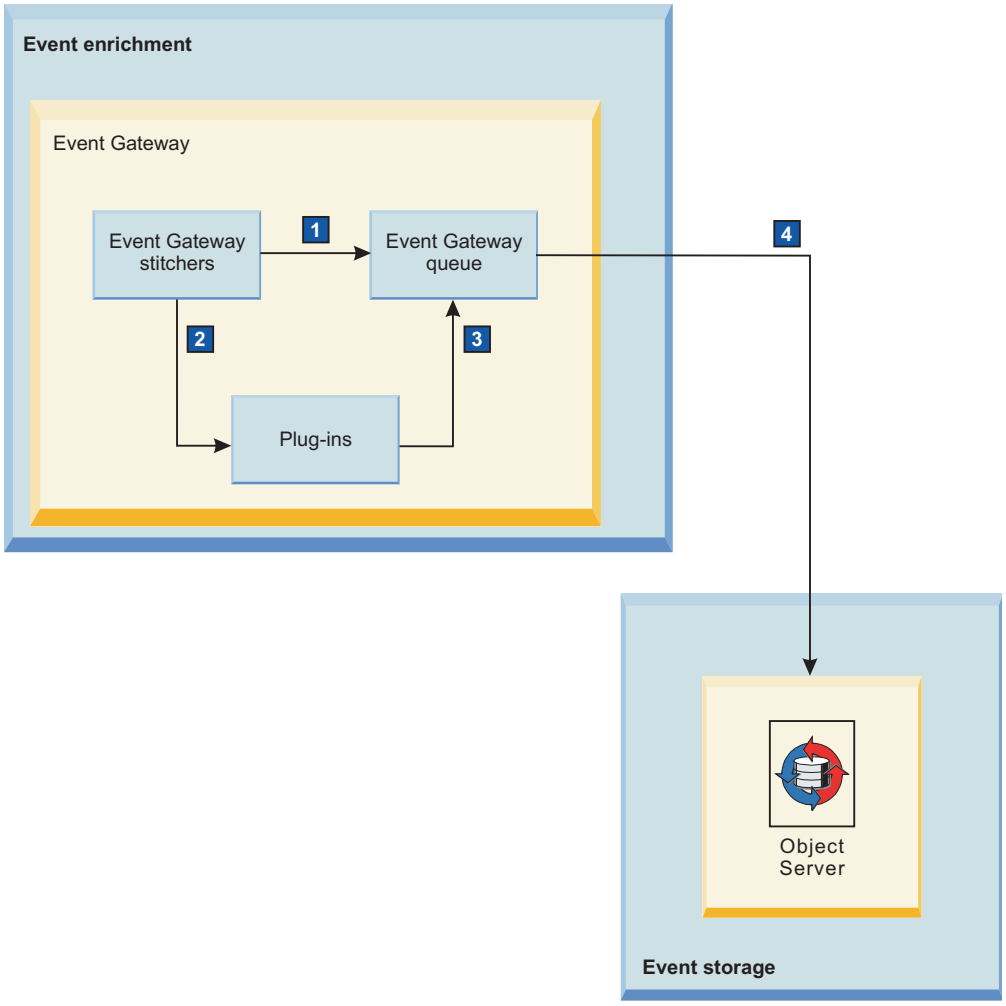


Figure 2. Outgoing Event Gateway queue

1 Standard enriched events placed on Event Gateway queue
Following standard event enrichment, the Event Gateway stitchers place enriched data on the Event Gateway queue.

2 Events passed to plug-ins
If the Event Gateway stitchers return a value of 1, then the related events are passed to the plug-ins for further enrichment. Plug-ins select the events to enrich based on the associated event map and event state.

3 Events enriched by the plug-ins are placed on Event Gateway queue
The plug-ins place the events that they have enriched on the Event Gateway queue.

4 Enriched events sent to the ObjectServer
Every 5 seconds enriched event data is sent to the ObjectServer. The interval of 5 seconds is configurable.

By default, the Event Gateway stitchers enrich the event by populating the NmosManagedStatus, NmosEntityId, NmosObjInst, and NmosDomainName fields. These events are placed on the Event Gateway queue and wait for the next update. Meanwhile, the event is passed to the RCA plugin. The RCA plugin enriches the event by populating the NmosSerial and NmosCauseType fields and then places these events on the Event Gateway queue. Both of these updates arrive on the

Event Gateway queue within one 5 second interval. Consequently rather than performing two upates, the Event Gateway is able to queue the data up so only one update of the ObjectServer is performed for all of these fields.

Related tasks:

"Configuring the ObjectServer update interval field" on page 157
You can configure the interval that the Event Gateway uses to queue event enrichment updates to the ObjectServer.

"Modifying event map subscriptions" on page 163
You can change the event maps that a plug-in subscribes to. For example, if you add a new event map and want the system to perform RCA on events handled by that event map, then you must add that event map to the subscription list for the RCA plug-in.

Event states

The Event Gateway assigns a state to the event based on the type of event, and based on the Severity and Tally fields in the event. The event state is one of the parameters used by event plug-ins when subscribing to events.

Related concepts:

"RCA stitcher descriptions" on page 138
Use this information to understand what each RCA stitcher does.

"RCA stitcher sequence" on page 136
The stitchers that are called by the RCA plug-in and the sequence in which stitchers are run to determine root cause.

Event types

For the purposes of the Event Gateway, event types fall into three broad categories: Problem, Resolution, and Information.

For the purposes of the Event Gateway, event types fall into the following categories:

Type = 1: Problem
The Event Gateway assigns one of a number of different states to problem events, based on the previous state, and based on the Severity and Tally fields in the event.

Type = 2: Resolution
Resolution events are immediately assigned the Resolution state.

Type =13: Information
Information events are immediately assigned the Information state.

The full list of event types defined in the alerts.status table is presented in the table below and is mapped to one of the three categories listed above.

Table 14. Transition labels

Value of Type field in alerts.status	Event type as understood by the Event Gateway
0: Type not set	Unknown
1: Problem	Problem
2: Resolution	Resolution
3: Netcool/Visionary problem	Problem

Table 14. Transition labels (continued)

Value of Type field in alerts.status	Event type as understood by the Event Gateway
4: Netcool/Visionary resolution	Resolution
7: Netcool/ISMs new alarm	Problem
8: Netcool/ISMs old alarm	Problem
11: More Severe	Problem
12: Less Severe	Problem
13: Information	Information

Event state diagram

The event state diagram shows possible event states and describes how transitions occur between these states based on the values in the alerts.status Severity and Tally fields. The diagram also shows how different event types are handled.

The event state diagram is shown below. Each event is assigned one of these states as it passes through the Event Gateway. Each state transition corresponds to an updated event received from the ObjectServer. The event states are shown with an associated color as follows:

- Red indicates an active problem state.
- Green indicates an active clear state.
- White indicates that this is not an active state.

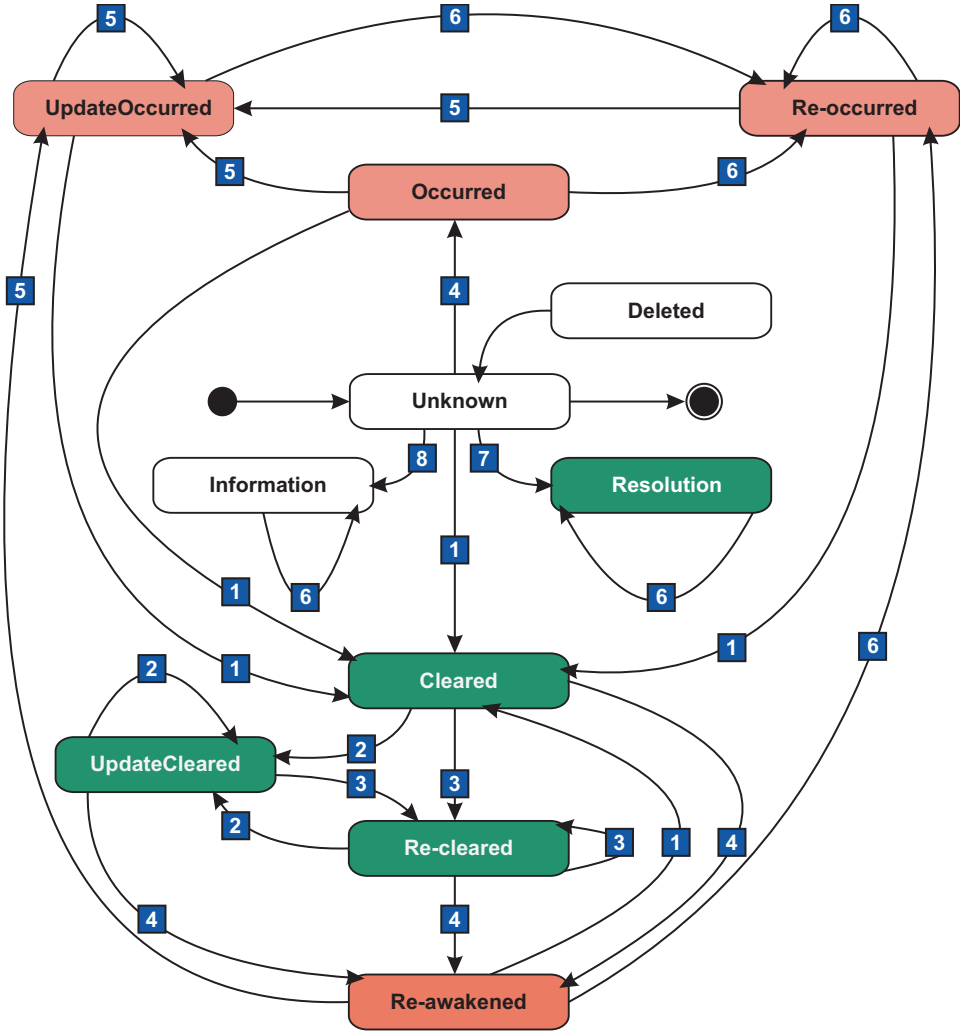


Figure 3. Event state diagram

Event state transitions

Each transition is labelled on the diagram with a number from 1 to 8. The following table lists the transitions associated with each label.

Table 15. Transition labels

Label	Field values in updated event
1	Severity =0
2	Severity = 0 Tally: Unchanged from previous event
3	Severity: 0 Tally: Changed from previous event
4	Severity: Non-zero
5	Severity: Non-zero Tally: Unchanged from previous event

Table 15. Transition labels (continued)

Label	Field values in updated event
6	Severity: Non-zero Tally: Changed from previous event
7	Type = 2: Resolution
8	Type = 13:Information

Event state descriptions

Event states are listed in the following table. All of the states listed refer to events of Type = 1 (Problem events), unless otherwise stated.

Table 16. Event states

State	Description
Cleared	The severity zero event was not previously known to the Event Gateway, or was in an active problem state.
Deleted	The event has been deleted from the ObjectServer. As this can happen at any time, this state can be reached from any other state except Unknown. Deletions are sent to plug-ins over the plug-in interface, and the state of the event in the Event Gateway becomes Unknown.
Information	Any incoming event that has the type Information (Type = 13) is given the Information state, irrespective of other field values.
Occurred	The non-zero severity event was not previously known to the Event Gateway.
Re-awakened	The non-zero severity event was previously known to the Event Gateway, but was not in an active problem state.
Re-cleared	The severity zero event was previously known to the Event Gateway, and has re-occurred.
Re-occurred	The non-zero severity event was previously known to the Event Gateway and a new occurrence of that problem event has occurred.
Re-synchronize	The Event Gateway has resynchronized with the ObjectServer. This is a synthetic state that does not correspond to any single ObjectServer event.
Resolution	Any incoming event that has the type Resolution (Type = 2) is given the Resolution state, irrespective of other field values.
Unknown	The event has not been detected by the Event Gateway. This is the initial and final state.
UpdateCleared	The severity zero event was previously known to the Event Gateway, and an update, as opposed to a reoccurrence, has been detected.
UpdateOccurred	The non-zero severity event was previously known to the Event Gateway, and an update, as opposed to a reoccurrence, has been detected.

Event handling

Event handling involves determining how to map each type of event from the ObjectServer to an entity in the topology data.

Event maps

Event handling is performed using event maps. The main function of an event map is to call a set of stitchers that perform topology lookup to determine the entity associated with the event and then enrich the event with topology data.

Event map selection:

The Event Gateway determines which event map to use based on the kind of event, as defined in the alerts.status EventId field. An example of a kind of event is an SNMP trap link down event.

Event map selection using the Event Gateway:

Use this information to understand how the Event Gateway is configured to select event maps to use for event handling.

If you choose to configure event map selection using the Event Gateway, then you must configure the Event Gateway config.precedence table. The config.precedence table is configured in the EventGatewaySchema.cfg configuration file. This file is located at: `$NCHOME/etc/precision/EventGatewaySchema.cfg`.

The following example shows how the config.precedence table is configured in the EventGatewaySchema.cfg configuration file.

The section of code that is relevant to selection of event maps to use for event handling is listed in the following lines from the example. This example insert configures the Event Gateway to use the event map LinkDownIfIndex for all events that have the EventId field set to SNMPTRAP-LinkDown. These are trap events originating from a Tivoli Netcool/OMNIbus probe.

The following table describes the relevant lines from this example:

Table 17. Lines of code relevant to the incoming event filter

Line numbers	Description
1	Configure the incoming filter by making an insert into the config.precedence table.
4-5	Specify an insert into the EventMapName and NcoEventIds field of the config.precedence table.
10	Set the EventMapName field to the value LinkDownIfIndex.
11	Set the NcoEventId field to the value of the EventId field in the alerts. status table; in this example, the event map LinkDownIfIndex is selected for all events where the alerts.status EventId value is SNMPTRAP-LinkDown.

```
1]    insert into config.precedence
2]    (
3]        Precedence,
4]        EventMapName,
5]        NcoEventId
6]    )
7]    values
8]    (
9]        910,
10]       "LinkDownIfIndex",
11]       "SNMPTRAP-LinkDown"
12]   );
```

Event map selection methods:

Selecting an event map to handle an event is done by specifying an eventPrecedence insert. An eventPrecedence insert can be specified by configuring any one of the following: Tivoli Netcool/OMNIbus probe rules files, IBM Tivoli Netcool/OMNIbus Knowledge Library, or the Event Gateway.

Selecting an event map to handle an event is done by specifying an eventPrecedence insert. An eventPrecedence insert can be specified by configuring one of the following:

Tivoli Netcool/OMNIbus probe rules files
: Configure the probe rules files to populate the NmosEventMap field of the alerts.status event record with the name of the event map and an optional precedence value. This is equivalent to an eventPrecedence insert.

IBM Tivoli Netcool/OMNIbus Knowledge Library
: Configure the Netcool/OMNIbus Knowledge Library configuration file, NcoGateInserts.cfg. This file contains a table where eventPrecedence inserts can be configured.

Event Gateway
: Populate the config.precedence table using the insert defined in the Event Gateway configuration file, EventGatewaySchema.cfg.

The eventPrecedence inserts configured in the Event Gateway override eventPrecedence inserts configured in either the Tivoli Netcool/OMNIbus probe rules files or in the Netcool/OMNIbus Knowledge Library. This enables you to locally override any eventPrecedence inserts configured in the network.

Default event maps:

Network Manager provides a default set of event maps. Use this information to understand which default event maps are available and what each event map does, and to understand how legacy event maps delegate to 3.9 event maps.

Default event maps

The following table describes the default event maps.

Table 18. Default event maps

Event map	Stitcher called by event map	Event map description
EntityFailure	LookupIp	Handles events where the LocalNodeAlias is sufficient to specify the entity, or if no further data is available. Expected input field: `LocalNodeAlias`, where this field contains the node IP address or DNS name.
EntityManagedStateChange	No related stitcher	Allows a plug-in to react to changes in the maintenance state of an entity (managed status changes to 0 or 1). This event map only applies to chassis or interfaces with an IP address. The event is not used by default. Expected input fields: `NmosEntityId` , where this field contains the NCIM entityId for the entity.
EntityMibTrap	LookupEntPhysEntry	Handles traps from the ENTITY MIB. Expected input fields: • `LocalNodeAlias` , where this field contains the node IP address, DNS name, sysName or entityName. • `LocalPriObj` , where this field contains an index from the entPhysicalTable; for example, `'entPhysicalEntry.2'`.
genericip-event	LookupMainNode	Process events that do not match any other event map. **Note:** This event map is intended as a catch-all. Plug-ins should not register interest in this eventMap. Instead, events of interest should be passed to the EntityFailure event map. Expected input field: `LocalNodeAlias`, where this field contains the node IP address or DNS name.
ItnmHealthChk	No related stitcher	Used by the Failover plug-in, to process Network Manager health check events. Expected input field: `Node`, where this field contains the name of the domain the health check is for.
ItnmLinkDownIfIndex	LookupIfEntry	Expects an interface event to be identified by the ifIndex if the interface `NmosEntityId` is not set. Expected input fields: • `LocalNodeAlias` , where this field contains the node IP address, DNS name, sysName or entityName. • `LocalPriObj` , where this field contains an `ifIndex` value from the `ifTable` table, in the format `ifEntry.`*`ifIndex`*, where *ifIndex* is the value of `ifIndex`; for example, `ifEntry.1`.
ItnmMonitorEventNoRca	LookupEntityId	Used to handle events raised by the Network Manager Polling engine, ncp_poller, for which root-cause analysis should not be performed. Expected input fields: `NmosEntityId` , where this field contains the NCIM entityId for the entity.

Table 18. Default event maps (continued)

Event map	Stitcher called by event map	Event map description
ItnmStatus	No related stitcher	Catch-all event map for Network Manager status information events that are not explicitly handled by any other event map, for example, ItnmHealthCheck. No action is taken for these events. For more information on Network Manager status information events, see the *IBM Tivoli Network Manager IP Edition Installation and Configuration Guide*.
LinkDownIfDescr	LookupIfEntry	Expects an interface event to be identified by the `ifDescr` value. Expected input fields: • `LocalNodeAlias` , where this field contains the node IP address, DNS name, sysName or entityName. • `LocalPriObj` , where this field contains an `ifDescr` value from the `ifTable` table, in the format `ifEntry.`*`ifDescr`*, where *ifDescr* is the value of `ifDescr`; for example, `ifEntry.FastEthernet0/1`.
LinkDownIfIndex	LookupIfEntry	Expects an interface event to be identified by the `ifIndex` value. Expected input fields: • `LocalNodeAlias` , where this field contains the node IP address, DNS name, sysName or entityName. • `LocalPriObj` , where this field contains an `ifIndex` value from the `ifTable` table, in the format `ifEntry.`*`ifIndex`*, where *ifIndex* is the value of `ifIndex`; for example, `ifEntry.1`.
LinkDownIfName	LookupIfEntry	Expects an interface event to be identified by the `ifName` value. Expected input fields: • `LocalNodeAlias` , where this field contains the node IP address, DNS name, sysName or entityName. • `LocalPriObj` , where this field contains an `ifName` value from the `ifTable` table, in the format `ifEntry.`*`ifName`*, where *ifName* is the value of `ifName`; for example, `ifEntry.Fa0/1`.
NbrFail	LookupNbrFailure	Handles OSPF, LDP and BGP adjacency change events. In addition to performing the requisite lookup, the LookupNbrFailure stitcher also adds a `RemoteNodeEntityId` value, used by the RCA plug-in. Expected input fields: • `LocalNodeAlias` , where this field contains the node IP address, DNS name, sysName or entityName. • `RemoteNodeAlias` , where this field contains the neighboring node IP address or DNS name. • `LocalPriObj` , where this field contains an `ifDescr` value in the format `ifEntry.`*`ifDescr`*, where *ifDescr* is the value of `ifDescr`; for example, `ifEntry.ifFastEthernet0/1`.

Table 18. Default event maps (continued)

Event map	Stitcher called by event map	Event map description
OspfIfState	LookupOspfIfEntry	Handles OSPF interface events. Expected input fields: • `LocalNodeAlias` , where this field contains the node IP address (used only for addresless interfaces). • `LocalPriObj` , where this field contains an index from the `ospfIfTable`; for example: – `ospfIfEntry.0.0.0.0.66` for addressless (IP unnumbered) interfaces – `ospfIfEntry.84.82.109.12.0` for serial or Ethernet interfaces
PollFailure	LookupIp	Covers events raised for a specific IP address, such as Tivoli Netcool/OMNIbus ping probe events. Expected input fields: `NmosEntityId` , where this field contains the NCIM entityId for the entity.
PrecisionMonitorEvent	LookupEntityId	Used to handle events raised by the ITNM poller, for which root-cause analysis must be performed. Expected input fields: `NmosEntityId` , where this field contains the NCIM entityId for the entity.
Reconfiguration	LookupMainNode	Allows partial discovery of devices to be launched automatically based on reboot events (events with event ID of NmosSnmpReboot), using the Disco plugin. Expected input field: `LocalNodeAlias`, where this field contains the node IP address or DNS name.

Legacy event maps

The following table lists the legacy event maps and, for each legacy event map, specifies which 3.9 event map is delegated to.

Table 19. Legacy event maps

Legacy event map	Handled by the following 3.9 event map
EntityIfDescr	LinkDownIfDescr
NbrFailIfDescr	NbrFail
NcpHealthChk	ItnmHealthChk
OSPFIfStateChange	OspfIfState
OSPFIfStateChangeIP	OspfIfState

How legacy event maps delegate to 3.9 event maps

This example explains how the NbrFailIfDescr legacy event map delegates to the NbrFail 3.9 event map.

1. An event is received that has an eventId listed in the config.precedence table, mapping to the NbrFailIfDescr eventMap.
2. The NbrFailIfDescr eventMap delegates to the NbrFail eventMap, using the HandledBy field.

3. The event is treated exactly as if it had been mapped to the NbrFail eventMap in the first place, that is it will be handled as follows:
   - The LookupNbrFailure stitcher is called. This stitcher is referenced in the `Stitcher` field relevant to the NbrFail event map entry in the config.eventMaps table.
   - The fields of the NbrFail eventMap (not the NbrFailIfDescr to which the event was originally mapped) are appended to the event before passing it to plug-ins.
   - Plugins that have registered interest in the NbrFail eventMap (not the NbrFailIfDescr to which the event was originally mapped) receive the event.

In this example, the flexibility of the stitcher language allows both types of event to be handled in the same way. If the ifDescr, expected by the legacy NbrFailIfDescr eventMap, is available, it is extracted and used within the LookupNbrFailure stitcher.

Related concepts:

Appendix F, "Network Manager event categories," on page 207
The events that are raised by Network Manager fall into two categories: events about the network being monitored and events about Network Manager processes.

Event Gateway stitchers

Event Gateway stitchers match events to an entity, perform a topology lookup, and then use the topology data retrieved to enrich the event data.

Event Gateway stitchers are stored in the following location: `$NCHOME/precision/eventGateway/stitchers/`.

For information on stitcher language, see the *IBM Tivoli Network Manager IP Edition Language Reference*.

There are four types of Event Gateway stitcher:

- "Topology lookup stitchers" on page 105
- "Data extraction stitchers" on page 109
- "Entity retrieval stitchers" on page 110
- "Event enrichment stitchers" on page 112

In addition, a number of Event Gateway stitchers are provided that are not used by default. These stitchers are provided as examples of extra functionality that can be added to the Event Gateway using stitchers. For more information, see "Stitchers not used by default" on page 114.

Stitcher selection using event maps:

Use this information to understand how the Event Gateway is configured to enable event maps to call specific Event Gateway stitchers.

Configuration of event maps to select specific Event Gateway stitchers is configured using the Event Gateway config.eventMaps table. The config.eventMaps table is configured in the EventGatewaySchema.cfg configuration file. This file is located at: `$NCHOME/etc/precision/EventGatewaySchema.cfg`.

The following example shows how part of the config.eventMaps table is configured in the EventGatewaySchema.cfg configuration file.This example insert configures the event maps listed in the following table to call the stitchers listed.

Table 20. Event maps and stitcher selected

Event map	Selected stitcher
PollFailure	IpLookup
ItnmMonitorEventNoRca	EntityIdLookup
PrecisionMonitorEvent	EntityIdLookup
LinkDownIfIndex	IfEntryLookup

The part of the code that is relevant to configuration of event maps to select specific Event Gateway stitchers is listed in the following lines from the example. The following table describes the relevant lines from this example. This code refers to the config.eventMaps table.

Table 21. Lines of code relevant to the incoming event filter

Line numbers	Description
1-10	Configure the event map PollFailure to select the stitcher IpLookup.
12-21	Configure the event map ItnmMonitorEventNoRca to select the stitcher EntityIdLookup.
23-32	Configure the event map PrecisionMonitorEvent to select the stitcher EntityIdLookup.
34-45	Configure the event map LinkDownIfIndex to select the stitcher IfEntryLookup. As this is a trap event, it is possible for the event to flap. Flapping is a condition where a device or interface connects to and then disconnects from the network repeatedly in a short space of time. This causes problem and clear events to be received one after the other for the same device or interface. Setting the EventCanFlap flag to 1 informs the RCA plug-in of this condition. The RCA plug-in checks if events with this flag set to 1 are actually flapping, that is the device or interface is continually going up and down, and if so, RCA waits until the event has settled down. Events that can flap are passed from the Event Gateway with an `EventCanFlap = 1` setting. These events are placed on the mojo.events database with `TimedEscalation = 1` and are left there for 30 seconds. After 30 seconds the TimedEventSuppression RCA stitcher processes all events that are at least 30 seconds old and have the `TimedEscalation = 1` setting.

```
1]    insert into config.eventMaps
2]    (
3]        EventMapName,
4]        Stitcher
5]    )
6]    values
7]    (
8]        "PollFailure",
9]        "IpLookup"
10]   );
11]
12]   insert into config.eventMaps
13]   (
14]       EventMapName,
15]       Stitcher
16]   )
17]   values
18]   (
19]       "ItnmMonitorEventNoRca",
20]       "EntityIdLookup"
21]   );
22]
23]   insert into config.eventMaps
24]   (
25]       EventMapName,
26]       Stitcher
27]   )
28]   values
29]   (
30]       "PrecisionMonitorEvent",
31]       "EntityIdLookup"
32]   );
33]
34]   insert into config.eventMaps
35]   (
36]       EventMapName,
37]       Stitcher,
38]       EventCanFlap
39]   )
40]   values
41]   (
42]       "LinkDownIfIndex",
43]       "IfEntryLookup",
44]       1
45]   );
```

Related reference:

"config.eventMaps Table" on page 233
The config.eventMaps table contains the event map that specifies how an event is processed. The table holds information specific to each type of Tivoli Netcool/OMNIbus event that is processed by the Event Gateway.

Event Gateway stitcher descriptions:

Event Gateway stitchers fall into four different categories. Stitchers in each category are responsible for different aspects of topology lookup and event enrichment.

Topology lookup stitchers:

These are the stitchers listed in the event maps. Topology lookup stitchers take the raw event, perform a topology lookup, and carry out some event enrichment. They frequently make use of other stitchers to perform tasks. For example, they might use data extraction stitchers to extract information from the event, entity retrieval stitchers to match the event to an entity in NCIM cache, and event enrichment stitchers to enrich the event.

The stitchers look up the topology in the NCIM cache that is broadcast by the Topology manager, ncp_model. For information on NCIM cache and on the format of data in NCIM cache, see the *IBM Tivoli Network Manager IP Edition Topology Database Reference*.

Topology lookup stitchers return a Boolean value of 1 or 0. These return values have the following meanings:

Return value 1
Pass the enriched event data to subscribing plug-ins. This usually means that the topology lookup was successful and an entity was found in the NCIM cache.

Return value 0
Do not pass event data to any plug-ins. This usually means that the topology lookup was not successful and no entity was found in the NCIM cache.

The following table describes the topology lookup stitchers.

Table 22. Topology lookup stitchers

Stitcher	Description	Expected arguments	Returns
LookupEntityId.stch	Looks up an entity based purely on the value of the NmosEntityId field of the event. This stitcher is intended for use only by events raised by the Network Manager Polling engine, ncp_poller. Performs default event enrichment based on the results of the lookup.	None. Called from event map.	Returns one of the following values: • 1: An entity is found in the NCIM cache. Pass the enriched event data to subscribing plug-ins. • 0: No entity is found in the NCIM cache. Do not pass the enriched event data to subscribing plug-ins.

Table 22. Topology lookup stitchers (continued)

Stitcher	Description	Expected arguments	Returns
LookupEntPhysEntry	Looks up an entity based on entPhysicalIndex data from the Entity MIB in the LocalPriObj field. Performs default event enrichment based on the results of the lookup.	None. Called from event map.	Returns one of the following values: • 1: An entity is found in the NCIM cache. Pass the enriched event data to subscribing plug-ins. • 0: No entity is found in the NCIM cache. Do not pass the enriched event data to subscribing plug-ins.
LookupIfEntry.stch	Looks up the index entry for an interface on a device based on field values in the event. This stitcher is used by events that occur on interfaces. Performs default event enrichment based on the results of the lookup.	None. Called from event map.	Returns one of the following values: • 1: An entity is found in the NCIM cache. Pass the enriched event data to subscribing plug-ins. • 0: No entity is found in the NCIM cache. Do not pass the enriched event data to subscribing plug-ins.
LookupIp.stch	Looks up an entity using an IP address or DNS name. Note that the entity that the stitcher finds can be an interface or a main node. Performs default event enrichment based on the results of the lookup.	None. Called from event map.	Returns one of the following values: • 1: An entity is found in the NCIM cache. Pass the enriched event data to subscribing plug-ins. • 0: No entity is found in the NCIM cache. Do not pass the enriched event data to subscribing plug-ins.
LookupMainNode.stch	Looks up a main node device. If a value is present in the NmosEntityId field of the event, then that value is used to determine the entity ID of the main node. Otherwise, fall back to the value in the LocalNodeAlias field. Performs default event enrichment based on the results of the lookup.	None. Called from event map.	Returns one of the following values: • 1: An entity is found in the NCIM cache. Pass the enriched event data to subscribing plug-ins. • 0: No entity is found in the NCIM cache. Do not pass the enriched event data to subscribing plug-ins.
LookupNbrFailure	Looks up an entity using an IP address or DNS name, along with an optional interface description. This stitcher also looks up the remote node that the event relates to. Performs default event enrichment based on the results of the lookup.	None. Called from event map.	Returns one of the following values: • 1: An entity is found in the NCIM cache. Pass the enriched event data to subscribing plug-ins. • 0: No entity is found in the NCIM cache. Do not pass the enriched event data to subscribing plug-ins.

Table 22. Topology lookup stitchers (continued)

Stitcher	Description	Expected arguments	Returns
LookupOspfIfEntr	Looks up an interface based on ospfIfEntry data. Performs default event enrichment based on the results of the lookup. This stitcher checks the OSPF data based on one of the following formats: **ospfIfEntry.0.0.0.0.***ifIndex* An example of this format is: `ospfIfEntry.0.0.0.0.66`. In this example the extracted interface index value is 66. This format applies to interface events from addressless interfaces, also known as IP unnumbered interfaces. The format is used by P2P serial port interfaces. **ospfIfEntry.***ipV4Adress***.0** An example of this format is: `ospfIfEntry.84.82.109.12.0` In this example the extracted interface index value is the IP address 84.82.109.12. This format applies to OSPF interface events on interfaces that have IP addresses assigned to them. The format is used by serial and Ethernet interfaces.	None. Called from event map.	Returns one of the following values: • 1: An entity is found in the NCIM cache. Pass the enriched event data to subscribing plug-ins. • 0: No entity is found in the NCIM cache. Do not pass the enriched event data to subscribing plug-ins.

Example: LookupIp.stch stitcher:

Use this topic to understand how topology lookup stitchers work.

The LookupIp.stch stitcher looks up an entity using an IP address or DNS name. The entity that the stitcher finds can be an interface or a main node. The following table describes the key elements of the stitcher.

Table 23. Line-by-line description of the LookupIp.stch stitcher

Line numbers	Description
3-7	There is no trigger for this stitcher. The stitcher is automatically called by the PollFailure and EntityFailure event maps. When calling the stitcher, these event maps provide the associated event as the in-scope record.
11	Create a record named entity to store the topology data associated with the event (the in-scope record).
13	Access the NmosEntityId field within the event and load the value of this field into the nmosEntityId variable.
14	Use the GwEntityData() stitcher rule to look up the entity details in NCIM cache, based on the value of the nmosEntityId variable. For more information on the GwEntityData() stitcher rule, and other Event Gateway stitcher rules, see the *IBM Tivoli Network Manager IP Edition Language Reference*.

Table 23. Line-by-line description of the LookupIp.stch stitcher (continued)

Line numbers	Description
16-19	If the NmosEntityId field is NULL, this means that this is the first occurrence of this event. Consequently, the event has never been through the Event Gateway and has never been enriched. As an alternative to the NmosEntityId, determine the identity of the affected entity using the LocalNodeAlias field in the event record. Then use the GwIpLookup() stitcher rule to look up the entity details in NCIM cache, based on the value of the LocalNodeAlias variable. For more information on the GwIpLookup() stitcher rule, and other Event Gateway stitcher rules, see the *IBM Tivoli Network Manager IP Edition Language Reference*.
21	Set the return value of the stitcher using the foundEntity variable. Initially set the value of this variable to 0; this assumes that no entity has been found.
22-25	If an entity has been found, call the StandardEventEnrichment.stch stitcher to perform event enrichment on the event using the entity data retrieved using the lookup. Set the return value of the stitcher to 1.
27	Pass the return value to the Event Gateway.

```
1]    UserDefinedStitcher
2]    {
3]        StitcherTrigger
4]        {
5]            // There is no trigger, as the eventMaps will automatically
6]            // call this with the event as the in-scope record
7]        }
8]
9]        StitcherRules
10]       {
11]           Record entity;
12]
13]           int nmosEntityId = eval(int, '&NmosEntityId');
14]           entity = GwEntityData( nmosEntityId );
15]
16]           if ( entity == NULL )
17]           {
18]               entity = GwIpLookupUsing( "LocalNodeAlias" );
19]           }
20]
21]           int foundEntity = 0;
22]           if ( entity <> NULL )
23]           {   ExecuteStitcher( "StandardEventEnrichment", entity );
24]               foundEntity = 1;
25]           }
26]
27]           SetReturnValue( foundEntity );
28]       }
29]   }
```

Data extraction stitchers:

The sole purpose of these stitchers is to take a single event data string in standard format, and parse the string to extract a single value, which is returned.

The following table describes the data extraction stitchers.

Table 24. Data extraction stitchers

Stitcher	Description	Returns
ExtractEntPhysIndex.stch	Attempts to extract a textual value representing an interface identifier from an input data string of the form "entPhysicalEntry.*string identifier*". Typically, this is used to extract the value from an event field such as LocalPriObj or LocalRootObj.	String representing an interface identifier, which is usually expected to be an `ifName`, `ifDescr` or `ifAlias`.
ExtractIfIndex.stch	Attempts to extract the integer value representing an interface identifier from an input data string of the form "ifEntry.*numerical identifier*". Typically, this is used to extract the ifIndex value from an event field such as LocalPriObj or LocalRootObj.	Integer index
ExtractIfString.stch	Attempts to extract a textual value representing an interface identifier from an input data string of the form "ifEntry.*string identifier*". Typically, this is used to extract the ifIndex value from an event field such as LocalPriObj or LocalRootObj.	Integer index

Example: ExtractIfString.stch stitcher:

Use this topic to understand how data extraction stitchers work.

The ExtractIfString.stch stitcher attempts to extract a textual interface identifier from an input argument of the form `ifEntry.`*`string_identifier`*, where *string_identifier* is the textual interface identifier . This method is usually used to extract the ifIndex value from an event field such as LocalPriObj or LocalRootObj.

Table 25. Line-by-line description of the ExtractIfString.stch stitcher

Line numbers	Description
3-11	This stitcher is invoked by another stitcher, usually a topology lookup stitcher.
15	Initialize the ifString variable to null. The ifString variable will hold the results of the textual interface identifier extraction operation.
17	Read the input argument from the invoking stitcher and load this into ifInputStr variable.

Table 25. Line-by-line description of the ExtractIfString.stch stitcher (continued)

Line numbers	Description
19	Specify a regular expression to use as part of the pattern matching and data extraction operation.
21-27	Perform the pattern matching and data extraction operation.
29	Pass the extracted string back to the invoking stitcher.

```
1]    UserDefinedStitcher
2]    {
3]        StitcherTrigger
4]        {
5]            //
6]            // Called from another stitcher using the syntax:
7]            //
8]            // text ifString = "";
9]            // ifString = ExecuteStitcher( 'ExtractIfString', myStringField );
10]           //
11]       }
12]
13]       StitcherRules
14]       {
15]           text ifString = "";
16]
17]           text ifInputStr = eval(text, '$ARG_1');
18]
19]           text stringMatch = "^ifEntry\.(\S+)";
20]
21]           int stringMatchCount = MatchPattern( ifInputStr, stringMatch );
22]
23]           // We only recognise a match if we matched once on the entire field
24]           if (stringMatchCount == 1 AND REGEX0 == ifInputStr )
25]           {
26]               ifString = eval(text, '$REGEX1');
27]           }
28]
29]           SetReturnValue( ifString );
30]       }
31]   }
```

Entity retrieval stitchers:

These stitchers take predefined data, typically extracted from the event, and try to retrieve a matching entity from the entityData table in the NCIM cache. The stitchers return the retrieved entityData record, if found, and NULL otherwise.

The following table describes the entity retrieval stitchers.

Table 26. Entity retrieval stitchers

Stitcher	Description	Returns
EntityFromEntPhysIndex.stch	Takes as input a main node name in text form and an `entPhysicalIndex` value in integer form and attempts to retrieve any type of entity which can have an `entPhysicalIndex` value, as given in the Entity MIB.	Any type of entity which can have an `entPhysicalIndex`, as given in ENTITY MIB
EntityFromIfIndex.stch	Takes as input a main node name in text form and an `ifIndex` value in integer form and attempts to retrieve an interface on the given main node with the given `ifIndex` value.	An interface on the given main node with the given `ifIndex`
EntityFromIfString.stch	Takes as input a main node name in text form and an `ifDescr` value, an `ifName` value or an `ifAlias` value in integer form and attempts to retrieve an interface on the given main node with an `ifDescr` value, `ifName` value, or `ifAlias` value matching the supplied string.	An interface on the given main node with an `ifDescr`, `ifName` or `ifAlias` value matching the supplied string

Example: EntityFromIfString.stch:

Use this topic to understand how entity retrieval stitchers work.

The EntityFromIfString.stch stitcher looks up an interface based on a main node entityName and a string, which is expected to be the ifName, ifDescr or ifAlias of the interface. This returns the interface entity, if found in NCIM cache.

Table 27. Line-by-line description of the EntityFromIfString.stch stitcher

Line numbers	Description
3-7	This stitcher is invoked by another stitcher, usually a topology lookup stitcher.
11-12	Read the input arguments from the invoking stitcher.
16-27	Set up an SQL query to retrieve an entity data record for an interface based on a main node entityName and a string, which is expected to be the ifName, ifDescr or ifAlias of the interface.
30	Use the RetrieveSingleOQL() stitcher rule to run the query and retrieve the entity data record for the interface. For more information on the RetrieveSingleOQL() stitcher rule, and other Event Gateway stitcher rules, see the *IBM Tivoli Network Manager IP Edition Language Reference*.
32	Pass the result of the entity lookup back to the invoking stitcher.

```
1]    UserDefinedStitcher
2]    {
3]        StitcherTrigger
4]        {
5]            // There is no trigger, as this is explicitly called from
6]            // other stitchers.
7]        }
8]
9]        StitcherRules
10]       {
11]           text mainNodeEntityName = ARG_1;
12]           text ifString = ARG_2;
13]
14]           Record entity;
15]
16]           text ifStringQuery =
17]                   "select * from ncimCache.entityData
18]                    where
19]                    ENTITYTYPE = 2
20]                    and
21]                    BASENAME = eval(text, '$mainNodeEntityName')
22]                    and
23]                    ( interface->IFNAME = eval(text, '$ifString')
24]                      or
25]                      interface->IFDESCR = eval(text, '$ifString')
26]                      or
27]                      interface->IFALIAS = eval(text, '$ifString') );";
28]
29]
30]           entity = RetrieveSingleOQL( ifStringQuery );
31]
32]           SetReturnValue( entity );
33]       }
34]   }
```

Event enrichment stitchers:

These stitchers enrich the event with the topology data retrieved by other stitchers.

The following table describes the event enrichment stitchers.

Table 28. Event enrichment stitchers

Stitcher	Description	Expected arguments	Returns
StandardEventEnrichment.stch	Performs default event enrichment by populating event fields that some plug-ins expect to use as well as fields that are fed back to update the event in the ObjectServer.	The entity that has been matched to the top-level in-scope event.	No return value.

Table 28. Event enrichment stitchers (continued)

Stitcher	Description	Expected arguments	Returns
EntityNotFound.stch	By default, the fields that are set by the Event Gateway are as follows: • NmosObjInst • NmosSerial • NmosCauseType This stitcher resets these basic fields if the event was previously assigned to this domain, but no matching entity is found.	None.	No return value.

Example: StandardEventEnrichment.stch:

Use this topic to understand how event enrichment stitchers work.

The StandardEventEnrichment.stch performs standard event enrichment. It populates event fields that plug-ins expect to be able to use (for example, entityType) as well as fields that are fed directly back from the Event Gateway to update the event in the ObjectServer. The only fields which are permitted to update the ObjectServer alerts.status table are those fields that are listed in the outgoing field filter, as defined in the in the FieldFilter section of the nco2ncp table within the EventGatewaySchema.cfg configuration file. For example, the entityType and entityName fields are added to the event for use by plug-ins, but do not actually enrich the event in the ObjectServer as these fields do not pass the outgoing field filter.

Table 29. Line-by-line description of the StandardEventEnrichment.stch stitcher

Line numbers	Description
3-8	This stitcher is invoked by another stitcher, usually a topology lookup stitcher.
12	Read in the entity data record from a topology lookup operation.
13	Declare a record to hold the fields to be used to enrich the event.
15-19	Initialize variables with values retrieved from the topology lookup operation.
19	Use the GwManagedStatus() stitcher rule to retrieve the managed status of the entity. For more information on the GwManagedStatus() stitcher rule, and other Event Gateway stitcher rules, see the *IBM Tivoli Network Manager IP Edition Language Reference*.
21-26	Set the field values in the record to be used to enrich the event.
28	Use the GwEnrichEvent() stitcher rule to update the fields in the event. For more information on the GwEnrichEvent() stitcher rule, and other Event Gateway stitcher rules, see the *IBM Tivoli Network Manager IP Edition Language Reference*.

```
1]    UserDefinedStitcher
2]    {
3]        StitcherTrigger
4]        {
5]            // There is no trigger. This is called from other stitchers
6]            // with the event as the in-scope record, and the entity as
7]            // the single argument.
8]        }
9]
10]       StitcherRules
11]       {
12]           Record entity = ARG_1;
13]           Record enrichedFields;
14]
15]           int entityType = @entity.entityData.ENTITYTYPE;
16]           text entityName = @entity.entityData.ENTITYNAME;
17]           int entityId = @entity.entityData.ENTITYID;
18]           int mainNodeId = @entity.entityData.MAINNODEENTITYID;
19]           int managedStatus = GwManagedStatus( entityId );
20]
21]           @enrichedFields.EntityType = entityType;
22]           @enrichedFields.EntityName = entityName;
23]           @enrichedFields.NmosDomainName = eval(text, '$DOMAIN_NAME');
24]           @enrichedFields.NmosEntityId = entityId;
25]           @enrichedFields.NmosManagedStatus = managedStatus;
26]           @enrichedFields.NmosObjInst = mainNodeId;
27]
28]           GwEnrichEvent( enrichedFields );
29]       }
30]   }
```

Related tasks:

"Example: Enriching an event with main node device location" on page 154
You can configure event enrichment so that the location of the main node device associated with an event is added to a field in the event.

"Example: Enriching an event with interface name" on page 155
You can configure event enrichment so that for all interface events, the name of the interface on which the event occurred is added to a field in the event.

Stitchers not used by default:

These stitchers are provided as examples of extra functionality that can be added to the Event Gateway using stitchers. These stitchers can be executed from other stitchers.

The following table describes the Event Gateway stitchers that are not used by default.

Table 30. Stitchers not used by default

Stitcher	Description	Expected arguments	Returns
EntityFromAtmIfDescr.stch	This stitcher illustrates how data extracted from an event (typically using the ExtractIfString stitcher) can be manipulated before looking up related topology in the NCIM cache. In this example, events are be raised with a shortened interface description, missing a standard suffix. That suffix is added before looking up the topology.	Main node name (text) `ifDecr` (text) with a missing `-atm subif` suffix	An interface on the given main node with an `ifDescr` matching the event interface description, concatenated with the predefined `-atm subif` string.
RetrieveAlertDetails.stch	Looks in the Object Server alert.details table for any data relating to the current in-scope event, and adds any data found to the in-scope event. Note that in this example, this is done indiscriminately and no filtering of data is performed. This also provides a template example of how to query the Object Server for additional data.	None	No return value

Example: Default enrichment of a Tivoli Netcool/OMNIbus trap event

Use this information to understand how a Tivoli Netcool/OMNIbus event is processed as it passes through the Event Gateway.

The Event Gateway receives a Tivoli Netcool/OMNIbus link down trap event from the ObjectServer. This event originates from a Tivoli Netcool/OMNIbus trap probe. This event therefore originates from outside of Network Manager.This example shows how the event is processed as it passes through the Event Gateway.

The steps in this process are as follows.

1. The Event Gateway receives a link down trap event from the ObjectServer. This event has the event ID `SNMPTRAP-LinkDown`.
2. The incoming event filter is applied to the event.

   This filter checks the LocalNodeAlias field of the event. The LocalNodeAlias field is not empty, and therefore the event passes the filter and moves to Step 3.
3. The Event Gateway assigns a state to the event based on the Severity, Tally, and Type fields in the event. The link down trap event has the following Severity and Tally information:

- Severity is non-zero
- Tally is 1
- Type is 'Problem'

Based in this information, the Event Gateway assigns the Occurred state to this event. This is a problem event and is a candidate for RCA.

4. The default incoming field filter is applied to the event. This field filter filters out alerts.status fields that do not participate in the Event Gateway processing and only allows the following fields through:
   - Acknowledged
   - AlertGroup
   - EventId
   - FirstOccurrence
   - LastOccurrence
   - LocalNodeAlias
   - LocalPriObj
   - LocalRootObj
   - Manager
   - NmosCauseType
   - NmosDomainName
   - NmosEntityId
   - NmosEventMap
   - NmosManagedStatus
   - NmosObjInst
   - NmosSerial
   - Node
   - RemoteNodeAlias
   - EventId
   - Serial
   - ServerName
   - Severity
   - Summary
   - SuppressEscl
   - Tally
   - Type
5. The Event Gateway determines how to handle this event, by determining which event map to use. Event maps define how to handle an event. At the same time a numerical precedence value is associated with the event.

   The event in this example has event ID `SNMPTRAP-LinkDown`. The insert that defines how to handle events with this event ID is defined in the Netcool/OMNIbus Knowledge Library configuration file, NcoGateInserts.cfg, and has the following form.

```
insert into config.precedence
(
    Precedence,
    EventMapName,
    NcoEventId
)
values
(
```

```
    910,
    "LinkDownIfIndex",
    "SNMPTRAP-LinkDown"
);
```

This insert instructs the Event Gateway to handle the SNMP link down trap event as follows:

- Apply the event map LinkDownIfIndex to the event.
  - This event map covers link up and link down events from the Tivoli Netcool/OMNIbus mttrapd probe These events all use the ifIndex value held in the LocalPriObj field of alerts.status to identify the interface from which this trap originated.
  - The RCA plug-in subscribes to events that are handled by the LinkDownIfIndex event map. Consequently this event will be passed to the RCA plug-in.
- When sending the event for RCA, use a precedence value of 910 for the event.

6. The Event Gateway uses the selected event map to determine which stitcher to call to perform a topology lookup.

   The selected event map is LinkDownIfIndex. In the EventGatewaySchema.cfg configuration file, the following insert is associated with this event map:

```
insert into config.eventMaps
(
    EventMapName,
    Stitcher,
    EventCanFlap
)
values
(
    "LinkDownIfIndex",
    "LookupIfEntry",
    1
);
```

   This insert instructs the Event Gateway to perform the following actions:

   - Use the stitcher LookupIfEntry to perform the topology lookup.

     The LookupIfEntry looks up the index entry for an interface on a device based on the field values in the event. Based on the value of the interface index extracted from the fields in the event, the stitcher retrieves a row from the NCIM cache consisting of entity and interface data. Another stitcher is called to perform event enrichment.
   - Set the EventCanFlap flag to 1 to inform the RCA plug-in that the related device or interface might be continuing going up and down.

7. Enriched event data is filtered by the outgoing field filter and placed on the Event Gateway queue.
8. The event is passed to the RCA plug-in.
9. Following further event enrichment by RCA, the event is placed on the Event Gateway queue.

Related concepts:

"Quick reference for event enrichment" on page 83
Use this information to understand how an event is processed as it passes through the Event Gateway.

Event Gateway plugins

Event Gateway plug-ins are modules of the Event Gateway process that receive enriched events from the Event Gateway and perform further event enrichment or take other action on these events.

Related concepts:

"Root-cause analysis" on page 130
Root cause analysis (RCA) is the process of determining the root cause of one or more device alerts. The root-cause analysis (RCA) plugin receives a subset of enriched events from the Event Gateway and determines which of the events are root cause and which events are symptoms. RCA only receives events that affect the routing of traffic through the network.

Plugin descriptions

Use this information to understand what each Event Gateway plugin does.

The following table describes the Event Gateway plug-ins. Each of these plug-ins receives enriched events from the Event Gateway and performs further enrichment of the event or takes some other action.

Note: The following plug-ins are enabled by default:

- Adaptive polling
- Failover
- RCA
- All Service-affected event (SAE) plug-ins

Table 31. Event Gateway plug-ins

Plug-in	Description
Adaptive polling	Writes a subset of related event and entity data to the ncmonitor.activeEvent table. This enables network views to be created based on event data (alert views). Poll policies are scoped based on network views and therefore polls can be defined based on network alerts. These are known as adaptive polls because polling is initiated based on network problem conditions.
Disco	Receives reboot events (traps) from the Event Gateway and initiates partial discovery based on these events. By default, this plug-in is only interested in events that indicate that a reboot has taken place. **Note:** By default partial discovery initiated by this plugin is limited to reboot events. Partial discovery is a resource-intensive process, and problems are likely to occur if partial discovery is triggered for all events received, especially those received in large quantities such as link downs.
Failover	Receives Network Manager health check (ItnmHealthChk) events from the Event Gateway and passes these events to the Virtual Domain process, which decides whether or not to initiate failover based on the event.
Root cause analysis (RCA)	Based on data in the event and based on the discovered topology, rules coded in RCA stitchers attempt to identify events that are caused by or cause other events.
Service-affected event (SAE)	By default, the SAE plug-in generates the following three types of SAEs: • MPLS VPN: generates service-affected events for MPLS VPNs • IP path: generates service-affected events for IP paths • Generic Network Manager service: can be configured to generated synthetic events when a an event occurs on a device associated with a custom service.

Table 31. Event Gateway plug-ins (continued)

Plug-in	Description
zNetView	Populates additional custom alerts.status fields that are used by IBM Tivoli NetView for z/OS®. **Note:** You must first add the following custom fields to the alerts.status table: • NmosClassName • NmosEntityType

Related concepts:

"Root-cause analysis" on page 130
Root cause analysis (RCA) is the process of determining the root cause of one or more device alerts. The root-cause analysis (RCA) plugin receives a subset of enriched events from the Event Gateway and determines which of the events are root cause and which events are symptoms. RCA only receives events that affect the routing of traffic through the network.

Related tasks:

"Enabling and disabling plug-ins" on page 161
Use the `ncp_gwplugins.pl` script to enable and disable plug-ins. Run the script separately for each plug-in.

"Listing plug-in information" on page 162
You can list information on Event Gateway plug-ins. For example, you can list the event maps and event states that each plug-in subscribes to.

"Modifying event map subscriptions" on page 163
You can change the event maps that a plug-in subscribes to. For example, if you add a new event map and want the system to perform RCA on events handled by that event map, then you must add that event map to the subscription list for the RCA plug-in.

"Setting plug-in configuration parameters" on page 165
You can set optional configuration parameters for the Event Gateway plug-ins using the `ncp_gwplugins.pl` script.

Chapter 8, "Managing adaptive polling," on page 51
Adaptive polls dynamically react to events on the network. You can create adaptive polls that manage a wide range of network problem scenarios.

Adaptive polling plug-in

Use this information to understand plug-in prerequisites, how the adaptive polling plug-in populates fields in the activeEvent table, as well as configuration details associated with the plug-in. The activeEvent table is in the NCMONITOR schema.

The adaptive polling plug-in removes rows from the activeEvent table when an event is cleared or deleted from the ObjectServer.

Required fields

This plug-in expects events supplied by the Event Gateway to be with populated with the following fields:

- Acknowledged
- AlertGroup
- EventId
- FirstOccurrence
- LastOccurrence
- LocalPriObj

- NmosCauseType
- NmosSerial
- NmosEntityId
- Serial
- Severity
- SuppressEscl
- Tally

Events in the activeEvent table

The activeEvent table contains only active problem events that have been matched to an entity in the topology and that meet the following conditions:

- Event is active. This means that the event has not been cleared, and is expressed in field terms by the relationship `Severity > 0`.
- Event is a problem event. The alerts.status field Type has the value `Problem`, `More Severe`, or `Less Severe`.
- Event has been matched to an entity. The Event Gateway has identified the main node. This is expressed in field terms by the relationship `NmosObjInst > 0`.

activeEvent table fields

The following table lists the fields in the activeEvent table that are populated by the adaptive polling plug-in. For an example of the creation of an alert view, which makes use of these fields, see the *IBM Tivoli Network Manager IP Edition Network Visualization Setup Guide*.

Table 32. Fields in the activeEvent table populated by the adaptive polling plug-in

Plug-in	Description
Acknowledged	Indicates whether the event has been acknowledged by the operator.
AlertGroup	Identifies the originator of the event.
domainMgrId	Unique integer that identifies the domain to which the affected device belongs.
entityId	The NmosEntityId of the event. The field is named entityId in this table for consistency with other NCIM topology database tables, thereby facilitating GUI functionality.
EventId	The name of the event type. Based on this field secondary polls can be initiated based on specific types of failure.
FirstOccurrence	The time in seconds (from midnight Jan 1, 1970) when this event was created or when polling started.
LastOccurrence	The time when this event was last updated.
LocalPriObj	The primary object referenced by the event. For use in managed object instance identification.
NmosCauseType	Stores the results of root cause analysis and allows root cause events to be identified.
NmosSerial	If the event was suppressed during root cause analysis, this field indicates the value in the Serial field of the suppressing event.
Serial	Unique identifier for the event, within the context of a single ObjectServer. This field is stored in the table in order to allow a key to be generated for the table.
ServerName	Unique name for the ObjectServer. In a system with multiple ObjectServers, this is required to uniquely identify an event. It exists in the table purely to allow a key to be generated for the table.

Table 32. Fields in the activeEvent table populated by the adaptive polling plug-in (continued)

Plug-in	Description
Severity	Severity of the event . **Note:** Severity zero events are not listed, as these events indicate that the alert has been resolved, and are therefore not required in alert views or in poll policies that use alert views as their scope.
SuppressEscl	Used to suppress or escalate the alert. The suppression level is manually selected by operators from the Active Event List.
Tally	A count of the number of occurrences of the event. This enables sporadic events such as one-off ping failures to be filtered out.

Configuration of the plug-in

At startup, or upon Event Gateway resynchronisation (a SIGHUP or at failover or failback), this plug-in will also populates the alertColors and alertConversions tables based on values in the Object Server alerts.colors and alerts.conversions table. The following parameters can optionally be set in the gwPluginConf table. This table is in the NCMONITOR schema.

Table 33. Optional configuration of the adaptive polling plug-in

Parameter name	Value	Purpose	Default
CopyAlertTablesAtStartup	Indicates whether to populate the alertColors and alertConversions tables. Possible values are: • True • False	This allows a single domain to populate these tables if problems occur when running multiple domains.	True
ActiveEventUpdateInterval	Interval, in seconds, at which the activeEvent table is updated.	Updates to the table are made in transactions, to try to minimise the load on the DB server. This interval identifies the period at which these transactoins are committed.	5

Related tasks:

Chapter 8, “Managing adaptive polling,” on page 51
Adaptive polls dynamically react to events on the network. You can create adaptive polls that manage a wide range of network problem scenarios.

Related reference:

“ncp_g_event plug-in database tables in ncmonitor” on page 241
Use this information to understand which Event Gateway configuration tables are available in the ncmonitor database and what type of information each table contains. Most of these tables relate to Event Gateway plug-in configuration.

Disco plug-in

Use this information to understand some basic information about how this plug-in operates, plug-in prerequisites, and configuration details associated with the plug-in.

Operation of the plug-in

The Disco plug-in subscribes to the Reconfiguration event map. By default, only events with event ID NmosSnmpReboot are handled by the Reconfiguration event map. These events are based on the Network Manager rebootDetection poll policy, and indicate that there has been a reboot on a device. To configure the Disco plug-in to handle other traps, configure the related event to be handled by the Reconfiguration event map.

The following diagram provides a brief summary of the operation of the Disco plug-in.

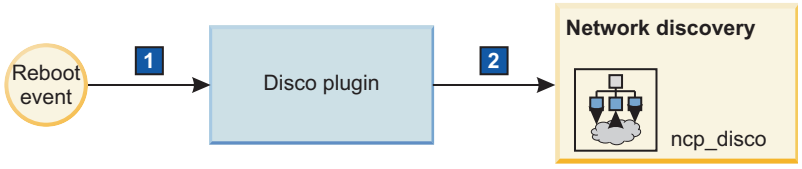


Figure 4. Operation of the Disco plug-in

1 Reboot event is received
A reboot event is passed from the Event Gateway to the Disco plugin

2 Partial discovery request is generated
The Disco plugin converts the reboot event into an appropriate OQL statement for the Discovery engine, ncp_disco, finders.returns table.

3 Rediscovery request sent to Discovery engine, ncp_disco
The OQL statement is run and this launches a partial discovery of the related device or devices.

Required fields

This plug-in expects events supplied by the Event Gateway to be with populated with the following fields:

- NmosObjInst

Configuration of the plug-in

The following parameter must be set in the ncmonitor.gwPluginConf table.

Table 34. Optional configuration of the adaptive polling plug-in

Parameter name	Value	Purpose	Default
StitcherSubDir	Name of the subdirectory within the `$NCHOME/precision/eventGateway/stitchers/` directory containing the stitchers for this plug-in.	Specifying the name of this directory allows only the Disco plug-in to parse its stitchers.	`Disco`

The following parameters can optionally be set in the ncmonitor.gwPluginConf table.

Table 35. Optional configuration of the adaptive polling plug-in

Parameter name	Value	Purpose	Default
StartupStitcher	Name of a stitcher in the subdirectory within `$NCHOME/precision/eventGateway/stitchers/`, to be run at initialization.	If a startup stitcher name is supplied then this stitcher will be run without arguments at startup, upon SIGHUP, or upon failover or failback.	None
SchemaFile	Name of a schema file in `$NCHOME/etc/precision/` to be parsed at initialization.	If a schema file name is supplied then this file will be parsed at startup, upon SIGHUP, or upon failover or failback, before any startup stitcher is run.	None

Related reference:

"ncp_g_event plug-in database tables in ncmonitor" on page 241
Use this information to understand which Event Gateway configuration tables are available in the ncmonitor database and what type of information each table contains. Most of these tables relate to Event Gateway plug-in configuration.

Failover plug-in

Use this information to understand plug-in operation as well as configuration details associated with the plug-in.

Operation of the plug-in

There is no configuration information associated with this plug-in.The following diagram provides a brief summary of the operation of the Failover plug-in.

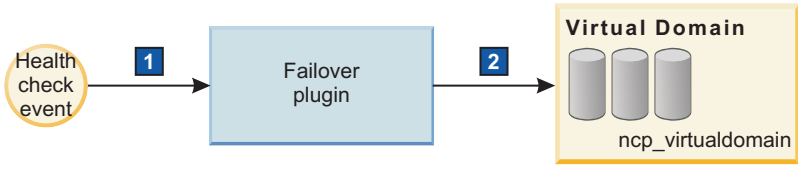


Figure 5. Operation of the Failover plug-in

1 Network Manager health check event is received
A Network Manager health check event is passed from the Event Gateway to the Failover plugin

2 Failover request is generated
The Failover plugin converts the Network Manager health check event into an appropriate OQL statement for the Virtual Domain, ncp_virtualdomain, state.domains table.

3 Failover request sent to Virtual Domain, ncp_virtualdomain
The OQL statement is run and this initiates failover or failback.

Required fields

This plug-in expects events supplied by the Event Gateway to have the Node field populated with the name of the affected Network Manager domain.

PostNCIMProcessing plug-in

Fix Pack 4

The PostNCIMProcessing plug-in runs any stitchers that are necessary after the NCIM database is updated. By default, the plug-in triggers the stitching of multiple domains into a single aggregation domain when a topology update event is received.

Operation of the plug-in

The PostNCIMProcessing plug-in subscribes to the ItnmStatus event map. The process flow for the plug-in is as described in the following steps:

1. An ItnmStatus event is received by the Event Gateway.
2. The Event Gateway calls the PostNCIMProcessing plug-in.
3. The PostNCIMProcessing plug-in runs the PostNcimProcessing stitcher, which checks the type of the event.
4. If the event is of type ItnmTopologyUpdate, and if cross-domain stitching is enabled, the PostNCIMProcessing stitcher runs the AggregationDomain stitcher.
5. The AggregationDomain stitcher checks that a discovery is not in progress, and then runs the other aggregation domain stitchers, which stitch the discovered domains together into one aggregation domain.

Required fields

This plug-in expects events that are supplied by the Event Gateway to be with populated with the following fields:

- NmosObjInst

Configuration of the plug-in

The following parameter must be set in the ncmonitor.gwPluginConf table.

Table 36. Required configuration of the PostNCIMProcessing plug-in

Parameter name	Value	Purpose	Default
StitcherSubDir	Name of the directory in `$NCHOME/precision/eventGateway/stitchers/` that contains the stitchers for this plug-in.	Specify the directory in which you want to put the NCIM processing stitchers, if you do not want to put the stitchers in the default directory, `PostNcimProcessing`. **Important:** Ensure that the directory exists and contains the stitchers. If the subdirectory does exist, the plug-in does not start.	`PostNcimProcessing`

The following parameters can optionally be set in the ncmonitor.gwPluginConf table.

Table 37. Optional configuration of the Disco plug-in

Parameter name	Value	Purpose	Default
StartupStitcher	Name of a stitcher in the subdirectory within `$NCHOME/precision/eventGateway/stitchers/` to be run at initialization.	If a startup stitcher name is supplied, this stitcher is run without arguments at startup, upon SIGHUP, or upon failover or failback.	None
SchemaFile	Name of a schema file in `$NCHOME/etc/precision/` to be parsed at initialization.	If a schema file name is supplied, this file is parsed at startup, upon SIGHUP, or upon failover or failback, before any startup stitcher is run.	None

SAE plug-in

The SAE plug-in generates service-affected events for MPLS VPNs and IP paths.

By default the SAE plug-in enables the system to generate the following types of service-affected event:

MPLS VPN service-affected events

A synthetic event generated when a Severity 5 (critical) fault event occurs on a provider edge (PE) or customer edge (CE) router or on any of the PE interfaces pointing towards a CE router in any of the discovered MPLS VPNs. The SAE generated is associated with the entity ID of the logical entity in the discovery that represents the collection of MPLS VPN devices on which the fault events occurred.

All PE to CE interfaces are added to a members list and an event on any of the interfaces in this members list causes the system to generate a synthetic MPLS VPN SAE.

You can enable the generation of SAE events based on interfaces dependencies deeper in the core network, by enabling the BGPPeerNextHopInterface discovery agent as part of your network discovery. This agent calls the AddLayer3VPNInterfaceDependency.stch stitcher.
This stitcher determines all PE to core provider router (P) interfaces and P to PE interfaces involved in a VPN. These PE -> P and P ->PE interfaces are added to a dependency list. An event on any of the interfaces in this dependency list causes the system to generate a synthetic MPLS VPN SAE. If an MPLS VPN SAE has already been generated based on an event on any of the interfaces in the members list, then any events in interfaces in the dependency list will be added as related events to that already generated MPLS VPN SAE.

For more information on the BGPPeerNextHopInterface discovery agent and the AddLayer3VPNInterfaceDependency.stch stitcher, see the *IBM Tivoli Network Manager IP Edition Discovery Guide*

IP path service-affected events

A synthetic event generated when an event occurs on a device within any of the IP paths created using the Network Paths GUI. The SAE generated is associated with the entity ID that corresponds to the IP path containing the device on which the event occurred.

Customizable service-affected events

The SAE plug-in can be configured to generated synthetic events when a an event occurs on a device associated with a custom service.

Fix Pack 3 In order to perform this configuration you must perform the following tasks:

1. Configure the Discovery engine, `ncp_disco`, to collect data for the custom service. Perform this configuration by writing a custom stitcher to define the custom service and ensure that this stitcher is called by the relevant standard Network Manager stitcher.

   Note: For help with writing custom stitchers, contact IBM Support.
2. Update NCIM cache to store data on the custom service in the `itnmService` NCIM database table.
3. Update the `config.serviceTypes` SAE plug-in database table to store data on the new custom service.

Fix Pack 3 For more information on `ncp_disco` and on the `itnmService` NCIM database table see the *IBM Tivoli Network Manager IP Edition Discovery Guide*.

Fix Pack 3 You can configure the SAE plug-in to generate more types of service-affected event. For example, you can configure the plug-in to create SAE events for MPLS VPN edge entities (one type of SAE) and for MPLS VPN core entities (another type of SAE). Use the SAE plug-in database tables used to configure the SAE plug-in.

Related tasks:

"Configuring the SAE plug-in" on page 166
Use this information to understand how to configure the SAE plug-in.

"Adding SAE types to the SAE plug-in" on page 167
You can configure the SAE plug-in to generate more SAE types than the three provided by default. For example, you can configure the plug-in to create SAE events for MPLS VPN edge entities (one type of SAE) and for MPLS VPN core entities (another type of SAE).

Related reference:

"SAE plug-in database" on page 239
The SAE plug-in database tables enable the SAE plug-into generate service-affected events for services such as MPLS VPNs and IP paths.

"config.serviceTypes table" on page 240
The config.serviceTypes table contains configuration information for the SAE plug-in.

zNetView plug-in

Use this information to understand plug-in prerequisites as well as configuration details associated with the plug-in.

Required fields

This plug-in expects events supplied by the Event Gateway to be with populated with the following fields:

- Acknowledged

The plug-in requires the following custom fields to exist in the alerts.status table.

Table 38. Optional configuration of the adaptive polling plug-in

Field name	Type	Description
NmosClassName	64-character string	Class of the device the event was raised against. This value is retrieved from the NCIM topology database chassis table.
NmosEntityType	32-bit integer	Type of entity the event is raised against. This value is specified in the NmosEntityId field.

Configuration of the plug-in

The following parameter must be set in the ncmonitor.gwPluginConf table.

Table 39. Optional configuration of the adaptive polling plug-in

Parameter name	Value	Purpose	Default
StitcherSubDir	Name of the subdirectory within the `$NCHOME/precision/eventGateway/stitchers/` directory containing the stitchers for this plug-in.	Specifying the name of this directory allows only the zNetView plug-in to parse its stitchers.	`zNetView`

The following parameters can optionally be set in the ncmonitor.gwPluginConf table.

Table 40. Optional configuration of the adaptive polling plug-in

Parameter name	Value	Purpose	Default
StartupStitcher	Name of a stitcher in the subdirectory within `$NCHOME/precision/eventGateway/stitchers/`, to be run at initialization.	If a startup stitcher name is supplied then this stitcher will be run without arguments at startup, upon SIGHUP, or upon failover or failback.	CheckAdditionalFields
SchemaFile	Name of a schema file in `$NCHOME/etc/precision/` to be parsed at initialization.	If a schema file name is supplied then this file will be parsed at startup, upon SIGHUP, or upon failover or failback, before any startup stitcher is run.	None

Related reference:

"ncp_g_event plug-in database tables in ncmonitor" on page 241
Use this information to understand which Event Gateway configuration tables are available in the ncmonitor database and what type of information each table contains. Most of these tables relate to Event Gateway plug-in configuration.

Plug-in subscriptions

Use this information to understand which event maps and event states each plugin is subscribed to.

Each Event Gateway plug-in subscribes to a different set of event maps.

Adaptive polling plug-in

The Adaptive polling plug-in subscribes to the following event maps:

1. EntityFailure
2. EntityIfDescr
3. EntityMibTrap
4. genericip-event
5. ItnmLinkDownIfIndex
6. ItnmMonitorEventNoRca
7. LinkDownIfDescr
8. LinkDownIfIndex
9. LinkDownIfName
10. NbrFail
11. NbrFailIfDescr
12. OSPFIfStateChange
13. OSPFIfStateChangeIP
14. PollFailure
15. PrecisionMonitorEvent

The Adaptive polling plug-in subscribes to the following event states:

1. Cleared
2. Deleted
3. Occurred
4. ReAwakened
5. ReOccurred
6. ReSync
7. Updated

Disco plug-in

The Disco plug-in subscribes to the following Reconfiguration event map.

Failover plug-in

This plug-in subscribes to the following event maps: ItnmHealthChkThe Failover plug-in subscribes to the following event states:

1. Cleared
2. Occurred
3. ReAwakened
4. ReCleared
5. ReOccurred
6. Resolution

7. ReSync

RCA plug-in

The RCA plug-in subscribes to the following event maps:

1. EntityFailure
2. EntityIfDescr
3. EntityMibTrap
4. ItnmLinkDownIfIndex
5. LinkDownIfDescr
6. LinkDownIfIndex
7. LinkDownIfName
8. NbrFail
9. NbrFailIfDescr
10. OSPFIfStateChange
11. OSPFIfStateChangeIP
12. PollFailure
13. PrecisionMonitorEvent

The RCA plug-in subscribes to the following event states:

1. Cleared
2. Deleted
3. Occurred
4. ReAwakened
5. ReOccurred
6. ReSync
7. Updated

SAE plug-in

The SAE plug-ins subscribe to the following event states:

MPLS VPN SAE plugin
This plug-in subscribes to the following event state: ReSync.

IP Path plugin
This plug-in subscribes to the following event state: ReSync.

ITNM Service plugin
This plug-in subscribes to the following event state: ReSync.

zNetView plug-in

The zNetView plug-in subscribes to the following event maps:

1. EntityIfDescr
2. EntityFailure
3. EntityMibTrap
4. genericip-event
5. ItnmLinkDownIfIndex
6. ItnmMonitorEventNoRca
7. LinkDownIfIndex

8. LinkDownIfDescr
9. LinkDownIfName
10. NbrFailIfDescr
11. NbrFail
12. OspfIfState
13. OSPFIfStateChange
14. OSPFIfStateChangeIP
15. PollFailure
16. PrecisionMonitorEvent
17. Reconfiguration

The zNetView plug-in subscribes to the following event states:

1. Information
2. Occurred
3. ReAwakened
4. ReOccurred
5. Resolution
6. ReSync
7. Updated

Root-cause analysis

Root cause analysis (RCA) is the process of determining the root cause of one or more device alerts. The root-cause analysis (RCA) plugin receives a subset of enriched events from the Event Gateway and determines which of the events are root cause and which events are symptoms. RCA only receives events that affect the routing of traffic through the network.

A failure situation on the network usually generates multiple alerts, because a failure condition on one device may render other devices inaccessible. The alerts generated indicate that all of the devices are inaccessible. Network Manager performs root cause analysis by correlating event information with topology information, thereby determining which devices are temporarily inaccessible due to other network failures. Alerts on devices which are temporarily inaccessible are suppressed, that is, shown as symptoms of the original root cause alert. Root cause alerts are shown in alert lists and topology maps; if the `severity_from_causetype` ObjectServer automation has been created and enabled, then these root cause alerts have the highest severity so that operators can easily identify them.

Related concepts:

"Event Gateway plugins" on page 118
Event Gateway plug-ins are modules of the Event Gateway process that receive enriched events from the Event Gateway and perform further event enrichment or take other action on these events.

Related reference:

"Plugin descriptions" on page 118
Use this information to understand what each Event Gateway plugin does.

Quick reference for RCA

Use this information to understand how an event is processed as it passes through the RCA plug-in.

The purpose of the RCA plugin is to determine, based on data in the event and on rules coded in RCA stitchers, to identify events that are caused by or cause other events. The steps are described in the following table.

Table 41. Quick reference for RCA

Action	Further information
1. An event is received from the Event Gateway. The RCA plug-in checks that this event meets its event maps and event state subscription requirements. In RCA terms this is known as a *trigger event* because it triggers RCA plug-in activity.	"Plug-in subscriptions" on page 128
2. The event is inserted into the RCA plug-in mojo.events database from where it can be retrieved for processing by the RCA stitchers.	"mojo.events events database table" on page 237
3. The event is passed to the ProcessEvent.stch, for root-cause analysis processing by the RCA stitchers.	"RCA stitchers" on page 136

Related tasks:

"Modifying event map subscriptions" on page 163
You can change the event maps that a plug-in subscribes to. For example, if you add a new event map and want the system to perform RCA on events handled by that event map, then you must add that event map to the subscription list for the RCA plug-in.

Precedence value

At the same time that an event map is selected to handle the event, a numerical precedence value is associated with an event. This precedence value is used by the RCA plugin in cases where there are multiple events on the same entity. The event with the highest precedence value on the entity is used to suppress other events.

A precedence value is configured for an event ID using the Event Gateway config.precedence table. The config.precedence table is configured in the EventGatewaySchema.cfg configuration file. This file is located at: `$NCHOME/etc/precision/EventGatewaySchema.cfg`.

The following example shows how the config.precedence table is configured in the EventGatewaySchema.cfg configuration file.

The section of code that is relevant to configuring a precedence value is listed in the following lines from the example. This example insert configures the Event Gateway to assign a precedence value of 910 to all events that have the EventId field set to SNMPTRAP-LinkDown. These are trap events originating from a Tivoli Netcool/OMNIbus probe. The code contains an insert to the config.precedence table.

The following table describes the relevant lines from this example:

Table 42. Lines of code relevant to the incoming event filter

Line numbers	Description
1	Configure the incoming filter by making an insert into the config.precedence table.
3	Specify an insert into the Precedence field of the config.precedence table.

Table 42. Lines of code relevant to the incoming event filter (continued)

Line numbers	Description
10	Set the Precedence field to the value 910.

```
1]    insert into config.precedence
2]    (
3]        Precedence,
4]        EventMapName,
5]        NcoEventId
6]    )
7]    values
8]    (
9]        910,
10]       "LinkDownIfIndex",
11]       "SNMPTRAP-LinkDown"
12]   );
```

Related reference:

"config.precedence table" on page 232
The config.precedence table lists events by event ID and contains the information necessary to determine which event has precedence when multiple events occur on the same interface. Based on the event ID, the config.precedence table also determines which event map to use to process an event from Tivoli Netcool/OMNIbus.

Default precedence values

By convention the Event Gateway assigns predefined threshold values that have special significance in the RCA plugin-in.

You can specify your own precedence values when configuring the Event Gateway.

You should specify a higher precedence value for events that meet either of the following conditions:

- Events on entities that are lower down the protocol stack. For example, confirmation that a physical port has failed would be higher precedence than an IP layer problem on that interface.
- Events that are a more certain indication of a problem. For example, contrast these two events: a ping fail event and a link down event. The less certain event is the ping fail. This might be because the ICMP packet could not reach the interface. That, in turn, could be due to a network problem between the polling station and the interface. The more certain event is an SNMP trap that explicitly states that a link has gone down because this is a more positive confirmation of a problem on the interface itself, or on its directly connected neighbour.

The following table lists the precedence values that the Event Gateway assigns by default.

Table 43. Default precedence values

Value	Meaning	Example events
0	Assigned to events that cannot cause other issues. During RCA, the event cannot suppress other events, but it can itself be suppressed.	SYSLOG-cisco-ios-SYS-CPUHOG SYSLOG-cisco-ios-BGP-NOTIFICATION

Table 43. Default precedence values (continued)

Value	Meaning	Example events
300	Reserved for non-authoritative events which suggest, but do not necessarily indicate, a failure on the device. For example, failure to reach a device does not necessarily indicate a problem on that device; this failure could be caused by a problem between the polling station and the device.	probeping-icmptimeout SNMPTRAP-IETF-OSPF-TRAP-MIB-ospfIfStateChange
600	Intended for protocol failures. Failures identified lower down the protocol stack should take higher precedence. For example, as OSPF runs over IP, an OSPF failure would be expected to have a lower precedence than an IP failure.	SNMPTRAP-IETF-OSPF-TRAP-MIB-ospfIfConfigError
900	Assigned to confirmed physical failures that indirectly imply a Link Down or Ping Fail (and most other events).	SNMPTRAP-cisco-CISCO-WIRELESS-IF-MIB-cwrTrapLinkQuality
910	Assigned to confirmed physical failures that directly indicate a Link Down or Ping Fail.	SNMPTRAP-linkDown SYSLOG-smc-switch-linkDown
10 000	Assigned to events that cannot be caused by other issues. During RCA, the event cannot be suppressed by other events, but it can become root-cause, suppressing other events.	SYSLOG-cisco-ios-CI-SHUTDOWN SNMPTRAP-riverstone-RIVERSTONE-NOTIFICATIONS-MIB-rsEnvirHotSwapOut

Poller entity

Use this information to understand what the poller entity is and how to configure it.

The poller entity, also known as the polling station, is the server from which Network Manager polls devices. If the polling station, usually the Network Manager server, is not within the scope of your network domain, then, to enable the RCA plug-in to perform isolated suppression, the IP address or DNS name of the ingress interface must be specified as the poller entity. This is the interface within the discovery scope from which network packets are transmitted to and from the polling station.

The poller entity is the name of a server to use to represent the local Network Manager server, and is stored in the config.defaults table in the field NcpServerEntity. A value is required in the NcpServerEntity field if the Network Manager server is not within the scope of the discovery.

The NcpServerEntity field must be configured as follows:

Table 44. NcpServerEntity field configuration settings

Is Network Manager server in discovery scope?	Value of NcpServerEntity field
Yes	Empty string
No	IP address or DNS name of the ingress interface

The following diagram shows the ingress interface (circled) when the Network Manager server is outside discovery scope.

Note: You must have run at least one discovery in order for the Event Gateway to find the poller entity in the NCIM database.

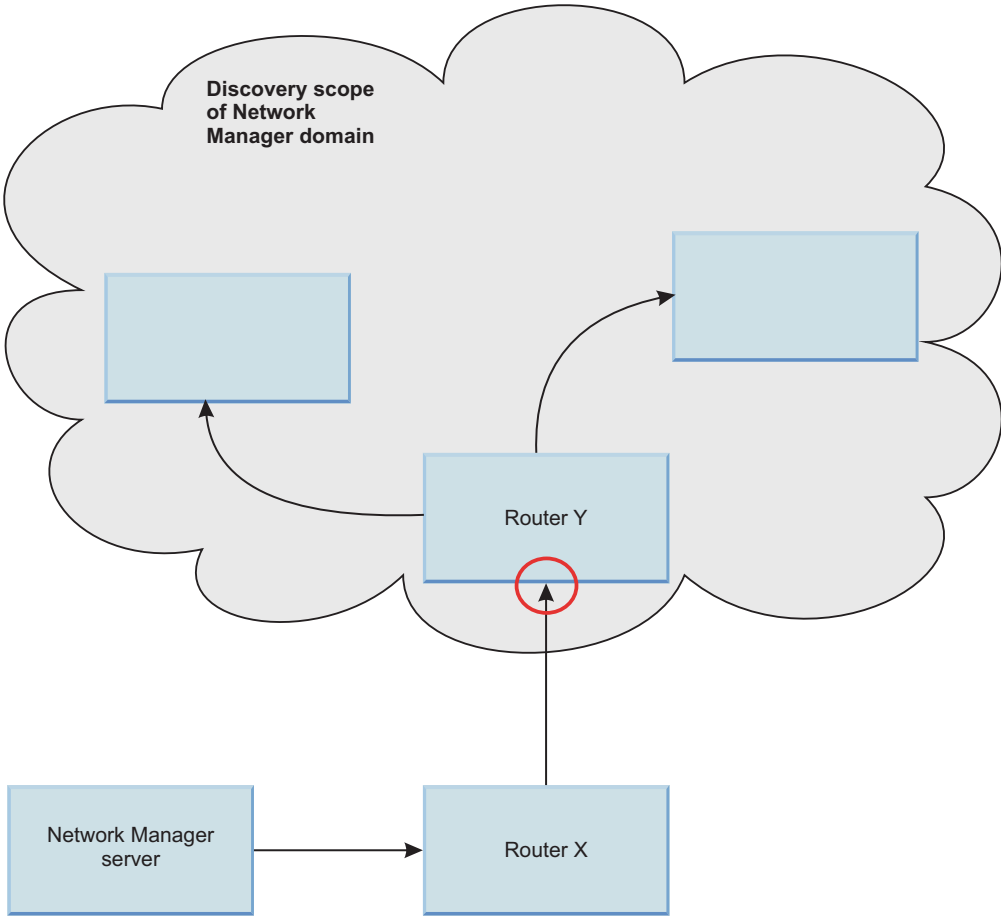


Figure 6. Ingress interface

Related tasks:

"Configuring the poller entity" on page 169
To enable the RCA plugin to perform isolated suppression when the Network Manager server is not within the scope of your network domain, specify the IP address or DNS name of the ingress interface as the poller entity.

RCA and unmanaged status

Use this information to understand how the RCA plug-in handles events from devices that are in unmanaged state, also known as maintenance state.

The method used by the RCA plug-in to handle events from devices that are in unmanaged state is governed by the value set for the `HonourManagedStatus` in the RCA plug-in configuration file `$NCHOME/etc/precision/RCASchema.cfg`. This field can take the following values.

- 1 (default value): instructs the RCA plug-in to honour the managed status of events. All events from unmanaged devices are ignored.

- 0: instructs the RCA plug-in to process events from unmanaged devices as normal events.

In each case, the RCA plug-in determines whether the device is unmanaged by looking at the NmosManagedStatus field of the event.

Assuming that the RCA plug-in is configured to honour the managed status of events (HonourManagedStatus = 1), then an event from an unmanaged device cannot be root cause and it cannot be suppressed.

If the event has an event state of ReOccurred and earlier occurrences of this same event indicated that the device was previously managed, then the event record in the mojo.events database is updated and will have its NmosCauseType, NmosSerial and SuppressionState reset to 0, in effect instructing the RCA plug-in to now ignore this event. The managed status of reoccurring events can change because of the following: managed Status is a property of an entity; for example, an interface. The managed status of the entity is stored in a field in the entity record. In addition, events raised on an entity also contain a field called NmosManagedStatus that records the managed status of the entity *at the time the event was raised*. Therefore it is possible for an event to occur when an entity is being managed, but then later on the same event could reoccur when the entity is unmanaged, that is, after the entity has changed state from managed to unmanaged.

The following scenarios explain how the RCA plug-in handles events whose managed status changes on subsequent occurrences.

Event changes from managed to unmanaged

The sequence is as follows:

1. The initially occurring event (for example, a ping fail event) is processed as normal for RCA, since the event had an NmosManagedStatus of 0, meaning that the entity was managed when the event first occurred.
2. Then, some time later, the interface entity is set to unmanaged; that is, the ManagedStatus value for the interface becomes 1.
3. The event on the interface reoccurs.
4. The reoccurred ping fail event now contains the field value NmosManagedStatus = 1, but the previous occurrence of this event, still in the database mojo.events, had the field value NmosManagedStatus = 0.
5. The RCA plug-in detects that the the value of the field NmosManagedStatus has changed from 0 to 1, for the ReOccurred (or Updated) event.
6. The RCA plug-in updates the event record in the database mojo.events and from then on treats the event is as it would treat a deleted event; that is, it reprocesses all the suppressees of the event as because this event no longer allowed to suppress events.

Event changes from unmanaged to managed

The sequence is as follows:

1. The initially occurring event (for example, a ping fail event) arrives with a NmosManagedStatus of 1 meaning that the entity was unmanaged when the event first occurred.Therefore the event is processed as if it were a deleted event and is not allowed to suppress events.
2. Then, some time later, the interface entity is set to managed; that is, the ManagedStatus value for the interface becomes 0.

3. The event on the interface reoccurs.
4. The reoccurred ping fail event now contains the field value `NmosManagedStatus` = 0, but the previous occurrence of this event, still in the database mojo.events, had the field value `NmosManagedStatus` = 1.
5. The RCA plug-in detects that the the value of the field `NmosManagedStatus` has changed from 1 to 0, for the ReOccurred (or Updated) event.
6. The RCA plug-in updates the event record in the database mojo.events and from then on treats the event is as a normal event; this event now allowed to suppress other events.

Related reference:

"config.defaults database table" on page 239
The config.defaults database table stores configuration data for the RCA plug-in event queue.

RCA stitchers

RCA stitchers process a trigger event as it passes through the RCA plug-in. Each RCA stitcher performs a specific task. A stitcher is called by the RCA plug-in and then the remaining stitchers are run in a specific order.

RCA plug-in stitchers are in `$NCHOME/precision/eventGateway/stitchers/RCA`.

For information about stitcher language, see the *IBM Tivoli Network Manager IP Edition Language Reference*.

RCA stitcher sequence

The stitchers that are called by the RCA plug-in and the sequence in which stitchers are run to determine root cause.

1. When the Event Gateway passes an event to the RCA plug-in, the `ProcessEvent.stch` stitcher is called to handle the event. This event is called the trigger event.
2. The `ProcessEvent.stch` stitcher determines which stitcher to call, depending on the event state of the trigger event, as described in the following table:

Event state of the trigger event	Stitcher that is called
Occurred ReAwakened, ReOccurred Resync Updated	`ProcessProblemEvent.stch`
Cleared Deleted	`ProcessResolutionEvent.stch` When an event is cleared or deleted, the RCA plug-in reprocesses any events that were suppressed by the cleared or deleted event.

3. 2 triggers are called by the `ProcessProblemEvent.stch` stitcher to attempt to suppress the trigger event. These triggers are as follows:
   - The `SuppressTrigger.stch` stitcher determines whether the trigger event can be suppressed by an existing event.
   - For OSPF or BGP events, the `PeerEntitySuppression.stch` stitcher determines whether the trigger event can be suppressed by another event on the peer entity.

If the SuppressTrigger.stch or the PeerEntitySuppression.stch stitchers cannot suppress the trigger event, the trigger event becomes root cause.

4. The RCA plug-in checks whether the trigger event is suppressed by another event on the same entity. That is, the check assesses if the trigger event is not the master event for the entity. If the trigger event is not the master event, it is prevented from suppressing other events, because the master event for the entity performs event suppression. In the subsequent steps, the ProcessProblemEvent.stch calls other stitchers, which attempt to use the trigger event to suppress other events. Each of the stitchers is run in turn.
5. The EntitySuppression.stch stitcher uses the trigger event to suppress other events by using contained entity principles. The event that has the highest precedence on the entity suppresses all the other events on that entity.
6. The ContainedEntitySuppression.stch stitcher uses the trigger event to attempt to suppress other events by using contained entity principles. The event on the containing entity suppresses events on all contained entities.
7. The IsolatedEntitySuppression.stch stitcher uses the trigger event to attempt to suppress other events by using downstream entity principles.
8. The ConnectedEntitySuppression.stch stitcher uses the trigger event to attempt to suppress other events using by connected entity principles. For example, when two interfaces are connected and there is an event on both, the event on one of the interfaces suppresses the event on the other interface.

Related concepts:

"Event states" on page 93
The Event Gateway assigns a state to the event based on the type of event, and based on the Severity and Tally fields in the event. The event state is one of the parameters used by event plug-ins when subscribing to events.

Predefined constants in RCA stitchers

Network Manager supplies predefined constants for RCA suppression type and RCA cause type. When coding new RCA stitchers or modifying existing RCA stitchers, you can store these predefined constants in an integer varible.

The predefined constants for suppression type are as follows:

$RCA_NO_SUPPRESSION
Not suppressed. A root cause event takes this state.

$RCA_ENTITY_SUPPRESSION
Suppressed by another event on the same entity.

$RCA_CONTAINED_SUPPRESSION
Contained suppressed; for example, failures on interfaces contained within a chassis device are suppressed by a failure on that chassis device.

$RCA_ISOLATED_SUPPRESSION
Isolated suppressed; failures on devices downstream of and isolated by a single chassis device are suppressed by a failure on that isolating chassis device.

$RCA_CONNECTED_SUPPRESSION
Suppressed by an event on a connected entity.

$RCA_PEER_SUPPRESSION
Peer entity suppression.

The predefined constants for cause type are as follows:

$RCA_UNKNOWN_CAUSE
Cause of event is unknown

$RCA_ROOT_CAUSE
Root cause event.

$RCA_SUPPRESSED
Suppressed event.

Note: Never use stitcher code set the causeType variable to $RCA_SUPPRESSED. This must only be done by the underlying RCA code.

RCA stitcher descriptions

Use this information to understand what each RCA stitcher does.

The following table describes the RCA stitchers.

Table 45. RCA stitchers

Stitcher	Description
ConnectedEntitySuppression.stch	Uses the trigger event to attempt to suppress other events using connected entity principles. For example, when two interfaces are connected and there is an event on both, the event on one of the interfaces suppresses the event on the other interface.
ContainedEntitySuppression.stch	Uses the trigger event to attempt to suppress other events using contained entity principles. The event on the containing entity suppresses events on all contained entities.
EntitySuppression.stch	Uses the trigger event to attempt to suppress other events using same entity suppression principles. The event with the highest precedence on the same entity suppresses the other events on that entity.
IsolatedEntitySuppression.stch	Uses the trigger event to attempt to suppress other events using downstream entity principles.
PeerEntitySuppression.stch	In the case of an OSPF or BGP event, determines whether the trigger event can be suppressed by an existing OSPF or BGP event.
ProcessEvent.stch	This is the head stitcher. It is called each time a trigger event is passed to the RCA plugin. The ProcessEvent.stch stitcher determines which stitcher to call based on the event state of the trigger event: • ProcessProblemEvent.stch is called to handle events with event states Occurred, ReAwakened, ReOccurred, Resync, and Updated. • ProcessResolutionEvent is called to handle events with event states Cleared and Deleted.
ProcessProblemEvent.stch	Handles problem events, that is, events with event states Occurred, ReAwakened, ReOccurred, Resync, and Updated. This stitcher calls the SuppressTrigger stitcher and the PeerEntitySuppression to try to suppress the trigger event using other events. It then calls the EntitySuppression, ContainedEntitySuppression, ConnectedEntitySuppression, and IsolatedEntitySuppression stitchers, in that order, to try to suppress other events using the trigger event.
ProcessResolutionEvent.stch	Handles resolution events, that is, events with event states Cleared and Deleted.

Table 45. RCA stitchers (continued)

Stitcher	Description
SuppressTrigger.stch	Determines whether the trigger event can be suppressed by an existing event.
TimedEventSuppression.stch	The purpose of this stitcher is to prevent the RCA plug-in from processing flapping events and thereby save resources. Events that can flap are passed from the Event Gateway with an `EventCanFlap = 1` setting. These events are placed on the mojo.events database with `TimedEscalation = 1` and are left there for 30 seconds. After 30 seconds the TimedEventSuppression RCA stitcher processes all events that are at least 30 seconds old and have the `TimedEscalation = 1` setting. **Note:** By waiting 30 seconds to process the event, the system ensures that the entity that generated the event has settled down and is not flapping. A flapping entity, for example, an interface that is generating a continuous stream of Link Down and Link Up events, might generate these events every two seconds. As the Link Up event passes through the RCA plug-in, the ProcessResolutionEvent stitcher will delete the Link Down event . Consequently, no flapping events will ever be processed by the TimedEventSuppression because they will already have been deleted during the 30 second wait time. Following processing, all events with a `TimedEscalation = 1` setting have the TimedEscalation field set to 2, to prevent any further processing.

Related concepts:

"Event states" on page 93
The Event Gateway assigns a state to the event based on the type of event, and based on the Severity and Tally fields in the event. The event state is one of the parameters used by event plug-ins when subscribing to events.

Examples of root cause analysis

These examples show how the RCA process performs root cause analysis based on consideration of different types of network devices and interfaces. The examples are for illustrative purposes only and are meant to show only the principles that RCA uses. RCA in larger networks is more complex.

The colors shown in the diagrams match the following event colors in the Active Event List:

- Red: root-cause event.
- Purple: symptom (suppressed) event.

For more information on identifying and investigating root-cause events in the Active Event List, see the *IBM Tivoli Network Manager IP Edition Network Troubleshooting Guide*.

Definition of downstream and upstream within RCA

Use this information to understand how the terms downstream and upstream are applied within the RCA plug-in.

Definition of terms

The terms downstream and upstream are used with reference to the poller entity.

Downstream
Specifies a location on the network topologically more distant from the polling station but on the same physical path as a second location.

Upstream
Specifies a location on the network topologically closer to the polling station but on the same physical path as a second location.

In complex networks, the distance of devices from the polling station changes as devices are deactivated. This change in distance has an impact on which devices are upstream or downstream.

Example

The figure below shows an example of upstream and downstream locations. In this example, device B is downstream of device A; therefore, device A is upstream of device B.

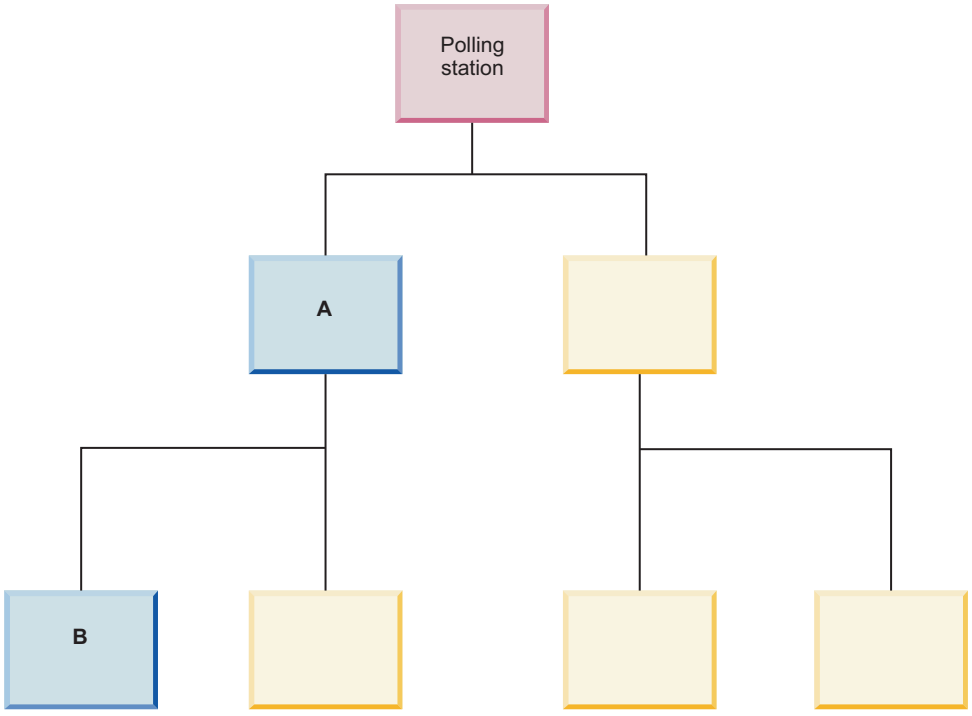


Figure 7. Downstream and upstream devices

Related reference:

"Isolated suppression of chassis devices" on page 144
A failure on a chassis device suppresses failures on all chassis devices isolated by the chassis device where the failure occurred. This is an example of *isolated suppression*.

"Isolated suppression for devices at the edge of a network" on page 147
A failure on a logical or physical interface that is the sole connection between other entities and the network suppresses failures in the downstream entities. This is an example of *isolated suppression*.

Chassis devices and loopback interfaces

In most cases, the RCA process assumes that if a chassis fails, then the root cause for other failures originates in the chassis. Chassis failures suppress failures on contained interfaces, connected interfaces and downstream chassis devices.

The loopback interface has a special function within a chassis device, whether router or switch. A loopback interface always has an IP address, which corresponds to the IP address of the chassis device.Network Manager IP Edition associates the loopback interface with the chassis during discovery. The loopback interface represents the whole chassis and can be polled individually. Failures on the loopback interface suppress failures on connected and contained entities in exactly the same way as failures on chassis devices.

Only events on chassis devices, interfaces, modules, and cards are allowed to connect-suppress other events. However, a chassis will not connect-suppress another chassis (or daughter card).

Contained interfaces:

A chassis failure suppresses all failures on interfaces contained within that chassis.

In the figure below, a failure on chassis device A suppresses failures on interfaces b, c and d. Interfaces b, c and d are all contained within chassis device A.

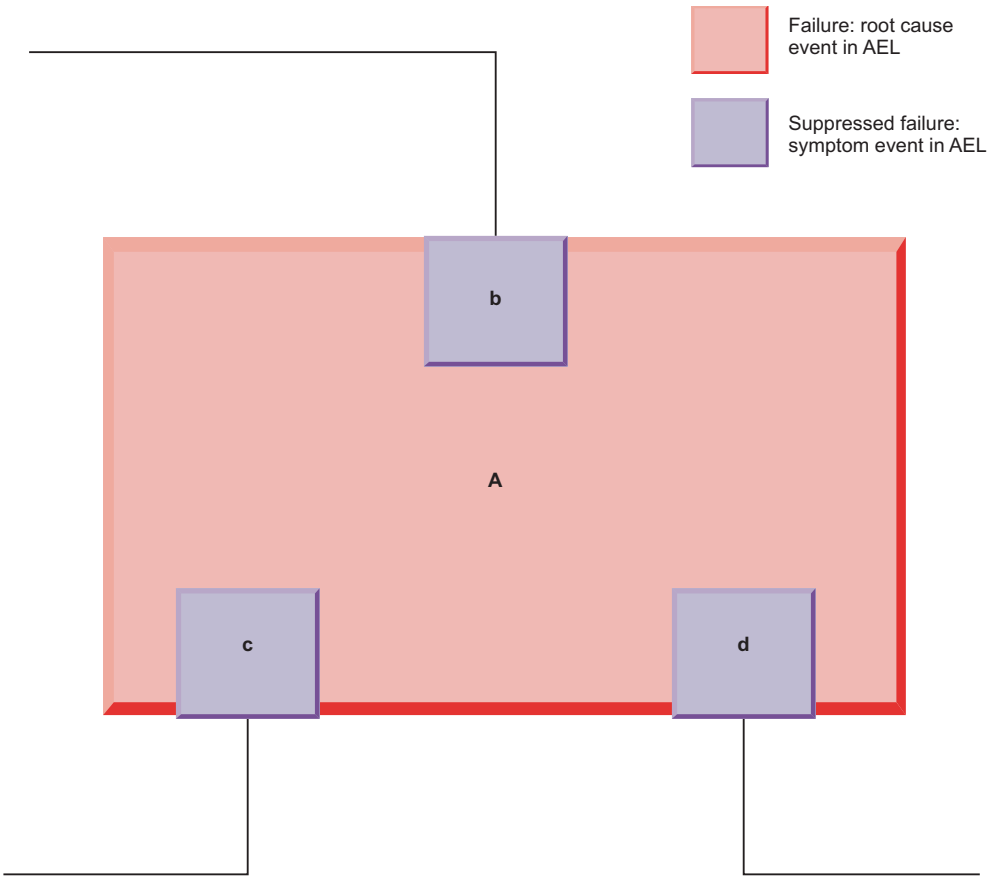


Figure 8. Chassis failure suppresses failures on contained interfaces

Connected interfaces:

A chassis failure suppresses all failures on interfaces connected to that chassis device. Failures are suppressed on both upstream and downstream interfaces as long as they are not isolation points.

In the figure below, device A suppresses failures on interfaces b, c, and d.

Note: If an interface is an isolation point in the graph, it cannot be connect-suppressed by an event on a neighboring entity.

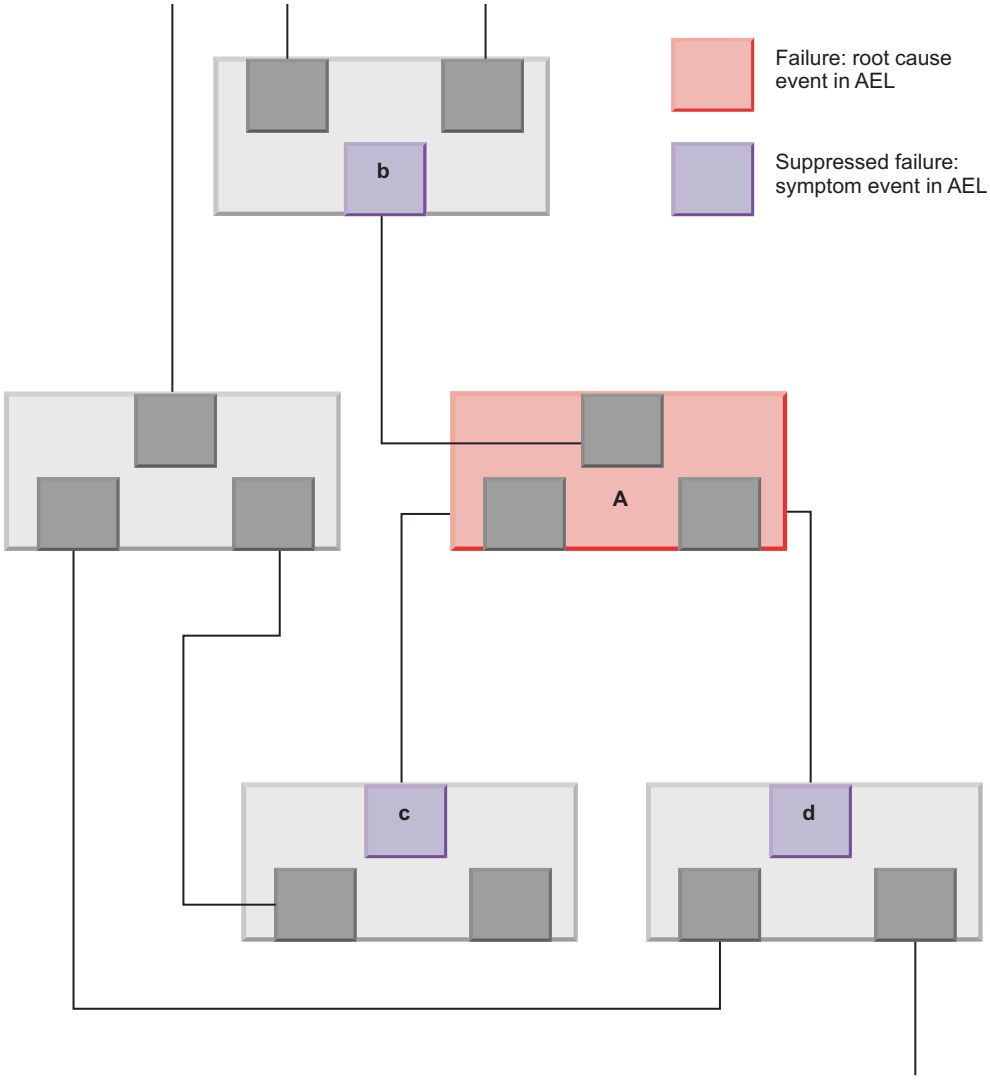


Figure 9. Chassis failure suppresses failures on connected interfaces

Entities connected to a contained entity:

A chassis device may contain one or more entities. Examples of entities which can be contained within a chassis device are VLANs, cards, and virtual routers. A contained entity, such as a card, may have one or more interfaces.

A failure on the chassis device suppresses failures on entities directly connected to any of the entities contained within that chassis device. In the figure below, entity B is contained within chassis device A. A failure on chassis device A suppresses a failure on interface d on device D and interface e on device E. Both interfaces d and e are directly connected to entity B.

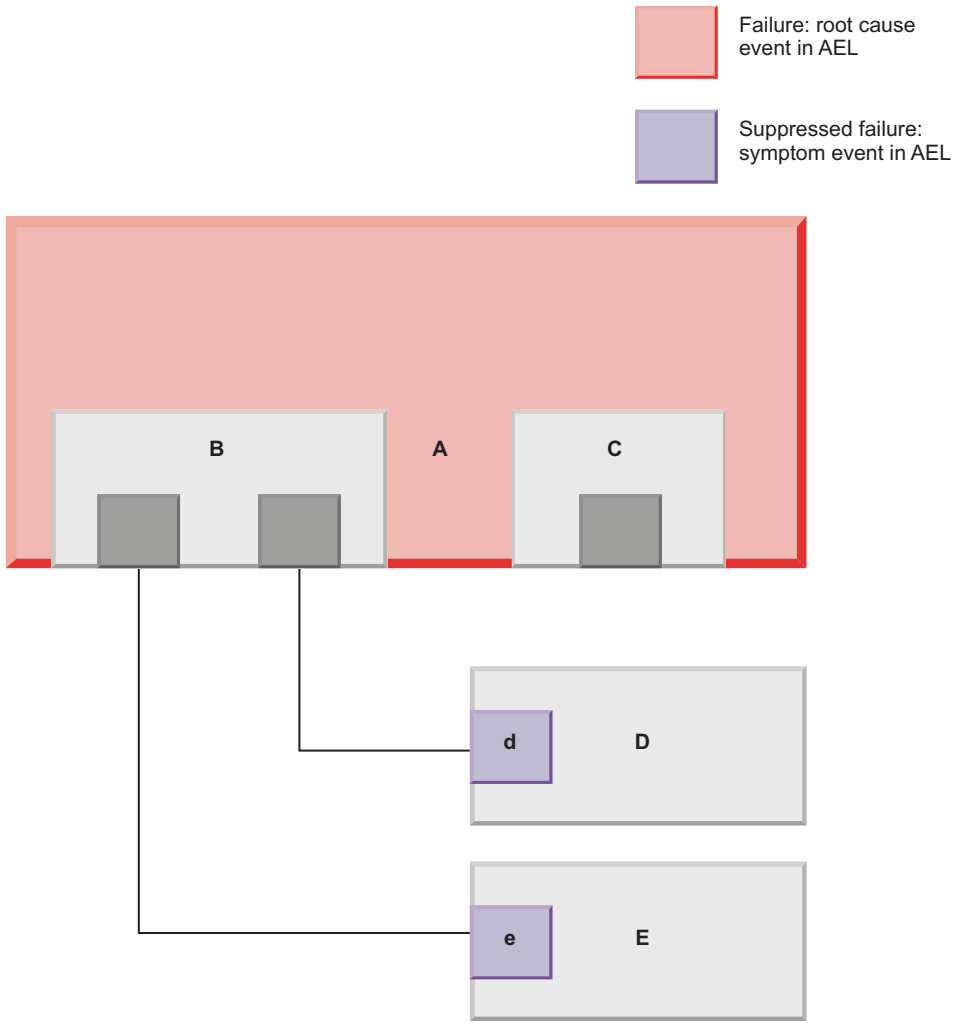


Figure 10. Chassis failure suppresses failures on devices connected to contained entities

Isolated suppression of chassis devices:

A failure on a chassis device suppresses failures on all chassis devices isolated by the chassis device where the failure occurred. This is an example of *isolated suppression*.

In the figure below, a failure on chassis device A suppresses failures on chassis devices B, C and D. Chassis devices B, C and D are all isolated by chassis device A.

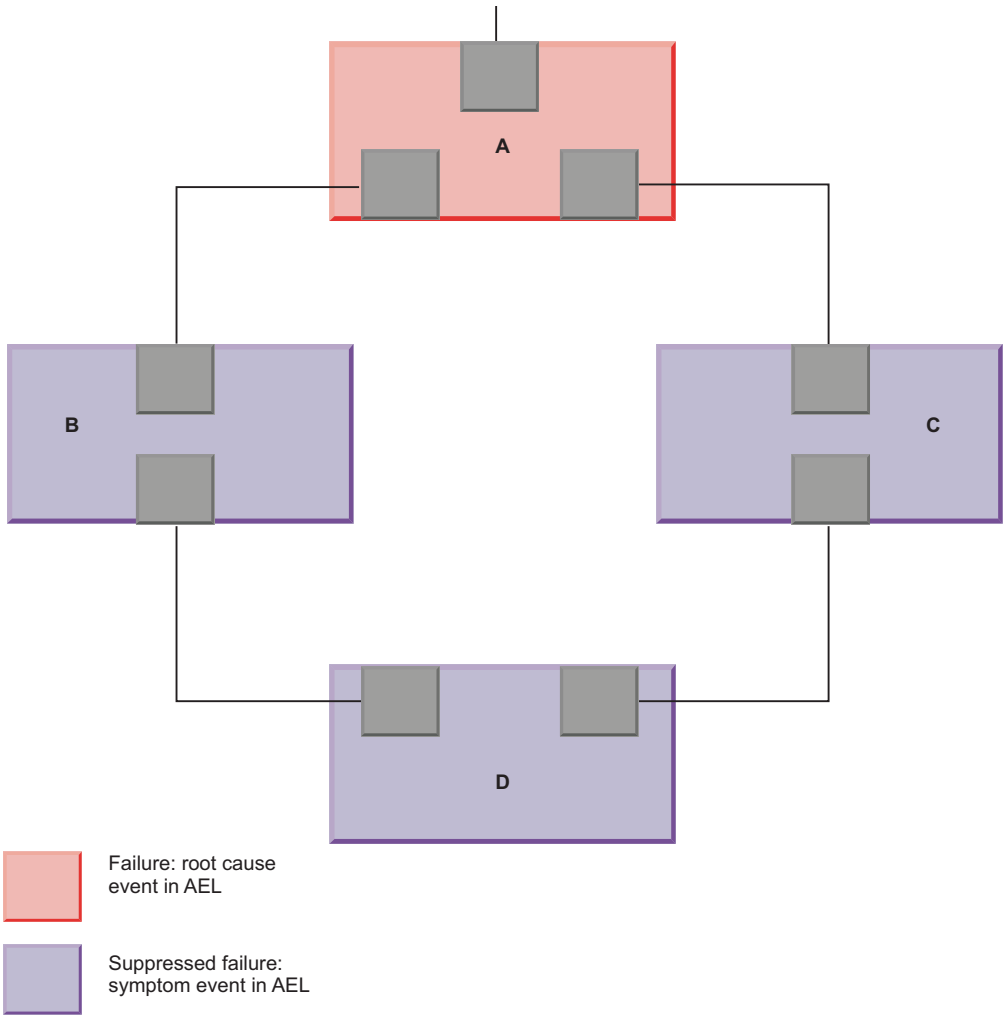


Figure 11. Chassis failure suppresses failures on downstream entities

Related reference:

"Definition of downstream and upstream within RCA" on page 140
Use this information to understand how the terms downstream and upstream are applied within the RCA plug-in.

Interfaces

If an interface is isolating downstream failures, then the interface failure can suppress the downstream failures.

A standard interface failure can suppress a second physical interface failure if the two interfaces are directly connected. The interface whose suppression rule fires first, suppresses the other interface. Suppression of one interface failure by a second interface failure can only occur if the interface failures are not already being suppressed by a chassis failure.

Note: If an interface is an isolation point in the graph, it cannot be connect-suppressed by an event on a neighboring entity.

A physical interface can contain multiple logical interfaces. A failure on a physical interface can suppress failures on its related logical interfaces. The physical interface can suppress its related logical interface even if there is connectivity

between the logical interface and an external neighbor. Even events on a suppressed physical interface can contain-suppress events on its associated logical interfaces.

In general, only events on chassis, interfaces, modules and cards are allowed to connect-suppress other events

Directly connected interface:

A standard physical interface failure suppresses a second physical interface failure if the two interfaces are directly connected.

The following constraints related to suppression of directly connected interfaces:

- A contained-suppressed interface cannot be connected suppressed.
- A suppressed interface can suppress connected interfaces.

In the figure below, failure on interface a suppresses the more recent failure on directly connected interface b.

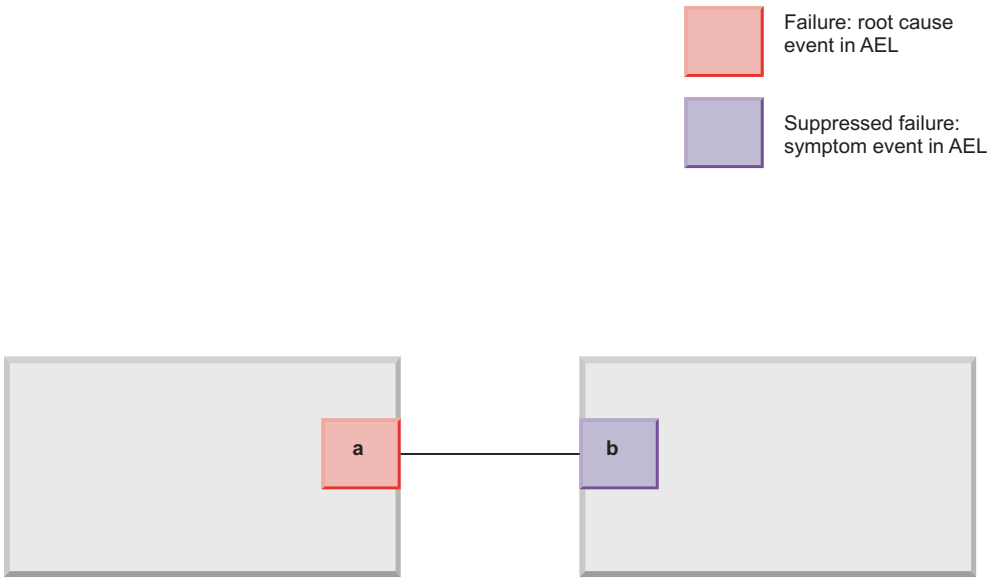


Figure 12. Interface failure suppresses more recent failure on directly connected neighbor interface

Related logical interface:

A failure on a physical interface suppresses failures on related logical interfaces.

In the figure below, failure on a physical interface suppresses failures on contained logical interfaces b and c.

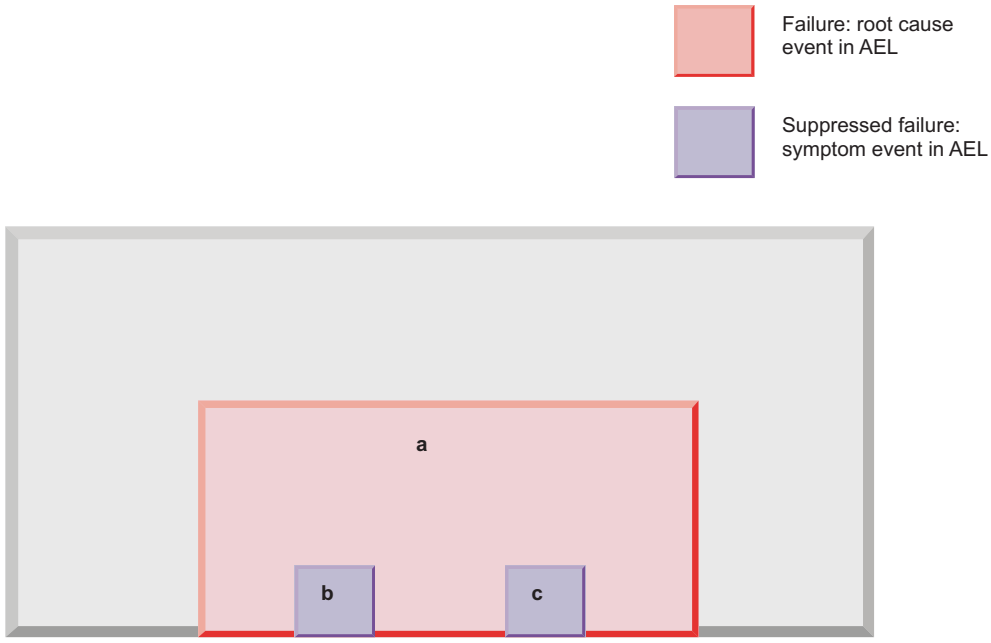


Figure 13. Physical interface failure suppresses failures on contained logical interfaces

Isolated suppression for devices at the edge of a network:

A failure on a logical or physical interface that is the sole connection between other entities and the network suppresses failures in the downstream entities. This is an example of *isolated suppression*.

In the figure below, failure on interface d in device A suppresses failures on devices B, C and D and their interfaces.

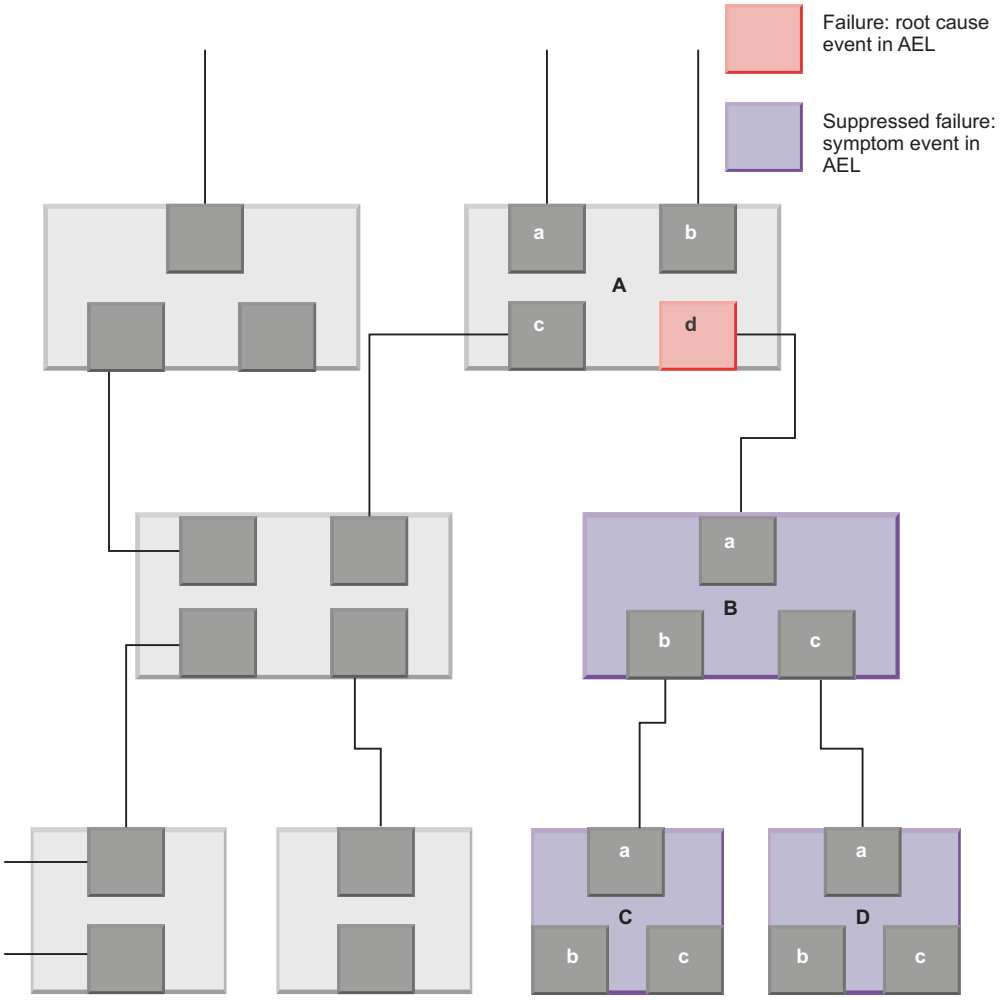


Figure 14. Interface failure suppresses more recent failure on directly connected neighbor interface

Related reference:

"Definition of downstream and upstream within RCA" on page 140
Use this information to understand how the terms downstream and upstream are applied within the RCA plug-in.

Checking topology paths used by RCA

Use the RCA path tool to check whether a topological path between network devices is available for the purposes of topological correlation.

About the RCA path tool

The RCA path tool provides a debugging aid for root cause analysis situations that involve isolated suppression.

The RCA path tool tells you the shortest path from A to Z, where A and Z are two nodes (for example, devices) in the topology. Use the RCA path tool to determine what paths exist in the topology and to determine where there are unexpected disconnects or unexpected additional paths, both of which will affect the root-cause analysis of the corresponding section of topology in the production environment.

The RCA path tool can display the following types of path between specified entities:

- **Full path**: this displays the shortest path between the source and target entities, regardless of the current state of the network. The full path does not change when events are placed on nodes in that path. For example, if there is an event, such as a PingFail alert on one or more of the intermediate devices along the path, this event is ignored. In RCA path tool queries, full path setting is indicated by the notation `atoz.full`.

  Note: To alter the full path, you would have to remove entities from the topology. This is a big change of course, and might be necessary on occasion for in-depth investigation.
- **Current path**: also known as the live path, this path displays the shortest path between the source and target entities, taking into account the current state of the network. For example, if there is an event, such as a PingFail alert on one or more of the intermediate devices along the shortest path to the target entity, then this path to the target entity is broken, and is not returned by the RCA path tool. If there is an alternative path, it is returned, even if it is a longer path to the target entity. In RCA path tool queries, current path setting is indicated by the notation `atoz.current`.

Note: The `atoz` database does not really exist, and neither do the `atoz.full` and `atoz.current` tables.

The most effective way to use the RCA path tool to perform debugging is to load your topology from cache and then to perform the following investigative activities on this cache topology:

1. Determine the existing path (current path) in the topology between two nodes of interest, A and Z.
2. Inject events onto specified nodes between A and Z along the path of interest:
   a. Does an alternative current path exist between A and Z?
   b. Is there no longer a path? If no current path exists, then the events that were injected will be isolated suppressed by an event on A or Z. Whether it is A or Z depends on the location of the poller entity. Assuming it is A then an event on A is a candidate for being the root cause suppressor of the events on the path between A and Z.

In order to inject events into devices in your topology cache, use the inject_fake_events.pl Perl script.For more information on the inject_fake_events.pl Perl script, see the *IBM Tivoli Network Manager IP Edition Administration Guide*.

Note: Do not confuse the RCA path tool with the path views tool that is available in the GUI. The RCA path tool is a command-line tool and is used primarily for troubleshooting root cause analysis; in contrast, the GUI-based path views tool provides graphical views to operators of devices and links that make up a network path between two selected devices.

Using the RCA path tool

Use these examples to understand how the RCA path tool can be used to display paths between specified source and target entities on the network.

Example of usage

The RCA path tool uses the OQL Service Provider, ncp_oql, to execute queries.For more information on the OQL Service Provider, see the *IBM Tivoli Network Manager IP Edition Administration Guide*.

The following example command queries the full path between a source device with an entityId field value of 6 and a target device with an entityId field value of 137.

```
ncp_oql -domain NCOMS -service Events -query "select * from atoz.full
where a = 6 and z = 137;"
```

The result of the query might look like this:

```
{
   ENTITYID=6;
   ENTITYNAME='router4';
}

{
   ENTITYID=385;
   ENTITYNAME='VLAN_OBJECT_router4_VLAN_37';
}

{  ENTITYID=137;
   ENTITYNAME='router4[ Fa0/3/3 ]';
}

( 3 record(s) :   Transaction complete )
```

Examples of queries

You can trace paths from a specific source entity, or optionally, from anything contained within that source entity. In addition, RCA path tool queries must always specify two entities, a source entity referred to as "a" and a target entity, referred to as "z", and these two entities represent the source and destination of the path. The source and target entities can be supplied as any combination of the following:

- Entity IDs
- Entity names
- IP addresses

Queries can additionally choose to allow a path to be traced from any entity contained by the source. This can be useful when dealing with VLANs, where a direct path from a containing chassis to an interface might not exist.

Note: Queries are logged in the trace file for the Event Gateway process, ncp_g_event, at debug level 1; for example, your query log file might be called ncp_g_event.NCOMS.trace.

The following queries provide examples of how you can use the RCA path tool.

1. Show the full path from entityId 102 to entityId 105.

   ```
   ncp_oql -domain NCOMS -service Events -query "select * from atoz.full
   where a = 102 and z = 105;"
   ```

2. Show the full path from the entity named 'rod' to the entity named 'freddy'.

```
ncp_oql -domain NCOMS -service Events -query "select * from atoz.full
where a = 'rod' and z = 'freddy';"
```

3. Show the current path from entityId 102 to the entity with IP address 172.21.226.3.

```
ncp_oql -domain NCOMS -service Events -query "select * from atoz.current
where a = 102 and z = '172.21.226.3';"
```

4. Show the current path from the interface named 'rod[0 [1]]' to entityId 105.

```
ncp_oql -domain NCOMS -service Events -query "select * from atoz.current
where a = 'rod[ 0 [ 1 ] ]' and z = 105;"
```

5. Show the full path from entityId 102 to the entity with IP address 172.21.226.3. If no path is found, try to find a path from anything contained by the container of entityId 102. In other words, go up one level in the container hierarchy to get the container identifier, and then try to construct the path using as source entity each one of the entities contained in that container.

```
ncp_oql -domain NCOMS -service Events -query "select * from atoz.full
where a = 102 and z = '172.21.226.3' and fromContained = 1;"
```

6. When there is no path, this will be clearly indicated by the output.

```
ncp_oql -domain NCOMS -service Events -query "select * from atoz.full
where a = 6 and z = 97;"
```

   If there is no path, the output will look something like the following:

```
{
    EntityId=0;
    EntityName='No path found from A to Z';
}
( 1 record(s) : Transaction complete )
```

Example: Determining potential root causes along a path

You can use the RCA path tool to simulate a failure along a network path. If there is no alternative path to the target entity, then the path to any device downstream of the failure will now be broken. In the production environment, the device corresponding to the failure device becomes root cause..

This example considers the devices A, B, C, and D, connected in a row. To keep things simple, interfaces are not shown:

```
A ------------ B ------------- C ------------- D
```

Network Manager polls device D from the poller entity. In the following diagram the poller entity is shown as entity X.

```
X
\
 \
  \
   \
    A ------------ B ------------- C ------------- D
```

If a PingFail alert is injected onto device B, this alert makes node B inactive and breaks the path from A to D and causes the RCA path tool to return the following results:

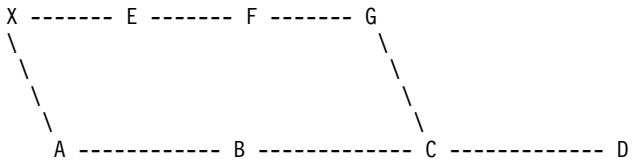
- **Full path (atoz.full)**: this displays the shortest path between nodes A and D, regardless of the current state of the network. Consequently, atoz.full displays the path from A to D.
- **Current path (atoz.current)**: this displays the shortest path between nodes A and D, taking into account the current state of the network. As node B is inactive, there is no path from A to D, therefore no path returned.

In the corresponding production environment, if a PingFail alert were to occur on node D, this alert would be suppressed, and the alert on node B would be highlighted as the root cause.

Example: Determining alternative paths

You can use the RCA path tool to determine alternative network paths in the case of a device failure. If there is an alternative path to the target entity, then in the production environment, the device corresponding to the failure device does not become root cause because there is an alternative path to the target entity.

This example considers a section of the network that includes two paths to node D, connected in a row. Network Manager polls device D from the poller entity. In the following diagram the poller entity is shown as entity X. To keep things simple, interfaces are not shown:



If a PingFail alert is injected onto device B, this alert makes node B inactive, and breaks the path from X to D via A. However, there is an alternative path from X to D, via E, and this causes the RCA path tool to return the following results:

- **Full path (atoz.full)**: this displays the shortest path between nodes X and D, regardless of the current state of the network. Consequently, this displays the path from X to D, via A, as this is the shortest path.
- **Current path (atoz.current)**: this displays the shortest path between nodes X and D, taking into account the current state of the network. Consequently, this displays the path from X to D, via E, as this is the current path. Node B is inactive, hence the path from X to D, via A is broken.

In the corresponding production environment, if a PingFail alert were to occur on node D, this alert would not be suppressed; also the alert on node B would not be shown as root cause.

Chapter 12. Configuring event enrichment

You can configure the way an event is processed as it passes through the Event Gateway.

Configuring extra event enrichment

You can configure the Event Gateway to perform extra event enrichment. The following examples illustrate the kinds of information that can be added to an event using event enrichment.

You can configure the Event Gateway to populate any field in the ObjectServer alerts.status table. You can populate an existing field or a customized field.

Note: The Event Gateway does not alter the alerts.status table. If you want to create a new field in the alerts.status table and have the Event Gateway populate this new field, you must first alter the alerts.status table in the ObjectServer to add the new field.

Modifications to the ObjectServer alerts.status table

The Event Gateway does not create new fields in the alerts.status table. If you are configuring extra event enrichment then you might need to configure the ObjectServer to add new fields to the alerts.status table.

The following examples describe typical custom event enrichment. Each example specifies whether any alerts.status table configuration is required prior to configuring the custom event enrichment.

Enriching a default event field that is not currently enriched
An example of this is where you want to enrich the PhysicalPort alerts.status field. This is a field that exists by default in the alerts.status table, and therefore there is no need to modify the ObjectServer.

Enriching a custom field that was already added earlier for a different purpose
An example of this is where you already have a field that is populated by one or more probes, and you want it populated for all events. In this example, some events that arrive via the monitor probe, from the poller, might have a populated EXTRAINFO_sysLocation field in the NCIM cache data. You have already added an NmosLocation field to the ObjectServer, and this field is populated from the monitor probe where possible. It can now be populated for all events. In this case there is no need to modify the ObjectServer.

Performing any topology enrichment from the NCIM topology database
In this case you want to enrich the event with any of the data from NCIM. You must first modify the ObjectServer to add the new field or fields to the alerts.status table.

© Copyright IBM Corp. 2006, 2016

Example: Enriching an event with main node device location

You can configure event enrichment so that the location of the main node device associated with an event is added to a field in the event.

Consider which field in the ObjectServer to populate. There already is a default Location field in the alerts.status table. This example assumes that you want to populate this field, unless it is already populated. If you have a reason to create a separate customized field to store the enriched location value, then you can add a field to the alerts.status table to store the main node device location; for example, NmosLocation. For information on how to add a custom field to an ObjectServer table, see the *IBM Tivoli Netcool/OMNIbus Administration Guide*.

The location of the main node device associated with an event is available in the NCIM topology database chassis table. This field can be accessed using NCIM cache, and is held in the ncimCache.entityData table.

For more information on the structure of NCIM cache tables and fields, see the *IBM Tivoli Network Manager IP Edition Topology Database Reference*.

The following steps explain how to configure this extra event enrichment.

1. Edit the Event Gateway schema file, `$NCHOME/etc/precision/EventGatewaySchema.cfg`, to allow the Event Gateway to update the new field. To do this, add the text in bold to the outgoing event filter. Remember to add a comma at the end of the line containing the `NmosSerial` field, before the line containing the new `Location` field.

```
insert into config.ncp2nco
(
    FieldFilter
)
values
(
    [
        "NmosCauseType",
        "NmosDomainName",
        "NmosEntityId",
        "NmosManagedStatus",
        "NmosObjInst",
        "NmosSerial",
        "Location"
    ]
);
```

   Note: Fields that are added to the outgoing event filter are automatically added to the incoming field filter, config.nco2ncp, thus ensuring that the current value of the field is retrieved. This allows the StandardEventEnichment stitcher in the next step to check the value of the InterfaceName field before updating it. This technique ensures that the Event Gateway does not keep updating the same value.

2. Edit the Event Gateway stitchers to retrieve the location information from the topology database and to populate the Location field. One way to do this is to add the following code to the StandardEventEnichment stitcher. Adding this code ensures that this procedure is performed for all topology events that are matched to an entity. Add this code to the stitcher immediately before the final line, the call to `GwEnrichEvent( enrichedFields )`. For more information on the `GwEnrichEvent()` stitcher rule, see the *IBM Tivoli Network Manager IP Edition Language Reference*.

Table 46. Lines of code relevant to the main node device location example

Line numbers	Description
1	Call the `GwMainNodeLookupUsing()` rule to ensure that chassis data is available for the current event. The event might have been raised on an interface, in which case the chassis data would not normally be available at this point. For more information on the `GwMainNodeLookupUsing()` stitcher rule, see the *IBM Tivoli Network Manager IP Edition Language Reference*.
5	Retrieve the sysLocation data from the chassis table. **Note:** When you retrieve data from NCIM cache, specify the field in the entity data in uppercase, for example, `@mainNode.chassis.SYSLOCATION`.
7 - 9	If the Location field is not already set then add the sysLocation data to the other fields to be enriched.

```
1]    Record mainNode = GwMainNodeLookupUsing( "LocalNodeAlias" );
2]
3]    if ( mainNode <> NULL )
4]    {
5]          text sysLocation = @mainNode.chassis.SYSLOCATION;
6]
7]          if ( sysLocation <> eval(text, '&Location') )
8]          {
9]               @enrichedFields.Location = sysLocation;      }
10]         }
```

Related concepts:

"Outgoing field filter" on page 90
The outgoing field filter defines the set of ObjectServer fields that may be updated by the Event Gateway.

Related reference:

"Example: StandardEventEnrichment.stch" on page 113
Use this topic to understand how event enrichment stitchers work.

Example: Enriching an event with interface name

You can configure event enrichment so that for all interface events, the name of the interface on which the event occurred is added to a field in the event.

You must create a new custom field in the ObjectServer alerts.status table to store the enriched interface name value. In this example, it is assumed that a new custom field called InterfaceName has been created in the alerts.status table. For information on how to add a custom field to an ObjectServer table, see the *IBM Tivoli Netcool/OMNIbus Administration Guide*.

The name of an interface is available in the NCIM topology database interface table. This field can be accessed using NCIM cache, and is held in the ncimCache.entityData table.

For more information on the structure of NCIM cache tables and fields, see the *IBM Tivoli Network Manager IP Edition Topology Database Reference*.

The following steps explain how to configure this extra event enrichment.

1. Edit the Event Gateway schema file, `$NCHOME/etc/precision/EventGatewaySchema.cfg`, to allow the Event Gateway to update the new field.

To do this, add the text in bold to the outgoing event filter. Remember to add a comma at the end of the line containing the `NmosSerial` field, before the line containing the new `InterfaceName` field.

```
insert into config.ncp2nco
(
    FieldFilter
)
values
(
    [
        "NmosCauseType",
        "NmosDomainName",
        "NmosEntityId",
        "NmosManagedStatus",
        "NmosObjInst",
        "NmosSerial",
        "InterfaceName"
    ]
);
```

Note: Fields that are added to the outgoing event filter are automatically added to the incoming field filter, config.nco2ncp, thus ensuring that the current value of the field is retrieved. This allows the StandardEventEnichment stitcher in the next step to check the value of the InterfaceName field before updating it. This technique ensures that the Event Gateway does not keep updating the same value.

2. Edit the Event Gateway stitchers to retrieve the interface name information from the topology database and to populate the InterfaceName field. One way to do this is to add the following code to the StandardEventEnichment stitcher. Adding this code ensures that this procedure is performed for all topology events that are matched to an entity. Add this code to the stitcher immediately before the final line, the call to `GwEnrichEvent( enrichedFields )` and after determining the entityType value. For more information on the `GwEnrichEvent()` stitcher rule, see the *IBM Tivoli Network Manager IP Edition Language Reference*.

Table 47. Lines of code relevant to the interface name example

Line numbers	Description
1	This event enrichment is only relevant for interface events. Check that this event relates to an interface by ensuring that the entityType value is 2, and if so, continue processing.
3	Retrieve the ifName data from the interface table. **Note:** When you retrieve data from NCIM cache, specify the field in the entity data in uppercase, for example, `@mainNode.chassis.SYSLOCATION`.
5 - 8	Only populate the InterfaceName field if the interface name value is not already present in the in-scope event.

```
1]    if ( entityType == 2 )
2]    {
3]         text interfaceName = @entity.interface.IFNAME;
4]
5]        if ( interfaceName <> eval(text, '&InterfaceName') )
6]        {
7]              @enrichedFields.InterfaceName = interfaceName;
8]        }
9]    }
```

Related concepts:

"Outgoing field filter" on page 90
The outgoing field filter defines the set of ObjectServer fields that may be updated by the Event Gateway.

Related reference:

"Example: StandardEventEnrichment.stch" on page 113
Use this topic to understand how event enrichment stitchers work.

Configuring the ObjectServer update interval field

You can configure the interval that the Event Gateway uses to queue event enrichment updates to the ObjectServer.

The default setting for the ObjectServer update interval is 5 seconds. You might want to alter this value to match the data flow on your system.

- Increase the value to group together more event enrichment updates in a single ObjectServer update. This decreases the load on the ObjectServer but increases the delay in event enrichment updates on the ObjectServer
- Decrease the value to speed up event enrichment updates to the ObjectServer. This increases the load on the ObjectServer, as it will have to manage more event enrichment updates.

The configuration file for the Event Gateway is the EventGatewaySchema.cfg configuration file. This file is located at: `$NCHOME/etc/precision/EventGatewaySchema.cfg`. The ObjectServer update interval is stored in the config.defaults table, in the field ObjectServerUpdateInterval.

1. Open the EventGatewaySchema.cfg configuration file.
2. Identify the insert statement into the config.defaults table. By default this insert statement has the following form:

```
insert into config.defaults
(
    IDUCFlushTime,
    ObjectServerUpdateInterval,
    NcpServerEntity
)
values
(
    5,
    5,
    ""
);
```

   By default the ObjectServerUpdateInterval field is set to 5 seconds.
3. Modify the value of the ObjectServerUpdateInterval field to the desired value, in seconds.

Related concepts:

"Outgoing Event Gateway queue" on page 91
The outgoing Event Gateway queue receives enriched events from the Event Gateway stitchers (main event enrichment) and from the plug-ins. In order to minimize the number of updates and hence minimize the load on the ObjectServer, updates to the Object Server are placed in a queue, aggregated, and sent to the ObjectServer at a specified interval. The default is 5 seconds.

Related reference:

"config.defaults table" on page 232
The config.defaults table contains general configuration data for the Event Gateway.

Using the OQL service provider to log into the Event Gateway databases

You must log into the databases using the object query language (OQL) service provider and the EventGateway service name to query the gateway databases.

The command-line example below logs in to the NcoGate service for the Event Gateway, which is running in the NCOMS domain.

```
ncp_oql -domain NCOMS -service EventGateway
```

User authentication for the OQL Service Provider is off by default. If authentication has been turned on, type a valid username and password at the prompt.

Querying the ObjectServer

You can use the OQL Service Provider to query the ObjectServer.

The OQL Service Provider command-line example below logs in to the ObjectServer service, which is running in the NCOMS domain on an ObjectServer called NCOMS.

```
ncp_oql -domain NCOMS -service ObjectServer -server NCOMS -username netcool
```

User authentication for the OQL Service Provider is off by default. If authentication has been turned on, type a valid username and password at the prompt.

Note: The `-server` argument is optional. If this argument is not specified then the server configured in the `$NCHOME/etc/precision/ConfigItnm.cfg` file is used.

Querying the NCIM database

You can use the OQL Service Provider to query the NCIM database.

The OQL Service Provider command-line example below logs in to the NCMONITOR schema within the NCIM service, which is running in the NCOMS domain. This is useful if you want to access a table in the NCMONITOR schema; for example, the activeEvent table.

```
ncp_oql -domain NCOMS -service Ncim -dbId NCMONITOR
```

User authentication for the OQL Service Provider is off by default. If authentication has been turned on, type a valid username and password at the prompt.

Note: The `-dbId` argument is optional.

Resynchronizing events with the ObjectServer

Issue the SIGHUP command to the Event Gateway to change the configuration of the Event Gateway.

Type this command: `kill -HUP` *`PID`*, where *`PID`* is the process ID of the Event Gateway.

The Event Gateway checks the timestamp on the configuration file. If the configuration file is modified, then the Event Gateway reads the configuration file again to process any configuration changes.

Note: This command also resynchronizes all events with the Event Gateway.

Processing steps for the SIGHUP command

The processing of the SIGHUP command is described in the following steps:

1. Event Gateway receives an HUP command.
2. Event Gateway stops listening for events on the ObjectServer IDUC channel.
3. Event Gateway empties its current cache of events. This cache is used to determine event state.
4. Event Gateway sends all its plug-ins a synthetic resync start event
5. RCA plug-in cleans out the mojo.events database table and redraws the graph based on data in NCIM cache.

   Note: The RCA plug-in does *not* reread the RCASchema.cfg configuration file or the RCA stitchers at this point.
6. Event Gateway retrieves all events from the Object Server, in the same way that it would at startup.
7. Event Gateway processes all events in the same way that it would at startup and passes any relevant events to the plug-ins.
8. Event Gateway sends a resync end event to its plug-ins.
9. Event Gateway resumes listening for events on the ObjectServer IDUC channel,

Related tasks:

Chapter 14, “Configuring root-cause analysis,” on page 169
You can configure the RCA plug-in.

Configuring common Event Gateway properties

You can configure common Event Gateway properties by editing the `NCP_G_EVENT.props` file.

Tivoli Netcool/OMNIbus gateways have a number of common properties and associated command-line options. Properties define settings for generic functions, such as message logging, for inter-process communication (IPC), and for common gateway settings, such as setting the timeout period that the client waits for the server to respond.

The Network Manager Event Gateway uses the default values for the Tivoli Netcool/OMNIbus gateway common properties. You can configure the Tivoli Netcool/OMNIbus gateway common properties to other values by editing the `NCP_G_EVENT.props` properties file, which is installed in the `$NCHOME/etc/precision` directory. For example, to avoid time-outs, you can specify a value other than the default (60 seconds) for the `Ipc.Timeout` common property to accommodate a quick drop in the connection to the Primary ObjectServer if no activity is detected.

Currently, `Ipc.Timeout` is the only Tivoli Netcool/OMNIbus gateway common property that you can configure. The `Ipc.Timeout` property specifies the time period (in seconds) that the client waits for the server to respond. If this time is exceeded, an error is logged. The default is `60` seconds.

To configure the `Ipc.Timeout` property:

1. Back up the `NCP_G_EVENT.props` properties file that was installed in the `$NCHOME/etc/precision` directory.
2. Open the `NCP_G_EVENT.props` properties file in a text editor.

3. Locate the `Ipc.Timeout` property and replace the default value of `60` seconds with a value appropriate for your network.
4. Save the `NCP_G_EVENT.props` properties file.

The following example configures the `Ipc.Timeout` property to the value 20 (seconds).

```
# INTEGER (IPC Session timeout), default 60 seconds
Ipc.Timeout: 20
```

Chapter 13. Configuring Event Gateway plug-ins

You can configure Event Gateway plug-ins. You can also view currently enabled plug-ins.

Enabling and disabling plug-ins

Use the `ncp_gwplugins.pl` script to enable and disable plug-ins. Run the script separately for each plug-in.

Run the `$NCHOME/precision/scripts/perl/scripts/ncp_gwplugins.pl` script. Use the command-line options to specify which plug-in and domain and to enable or disable a plug-in. The command-line options are described in the following table.

Table 48. `ncp_gwplugins.pl` command-line options

Command-line option	Description
`-domain` *DomainName*	Required: The name of a domain related to the plug-in. This domain is used to enable the script to read the relevant `DbLogins.cfg` file in order to connect to and update the relevant Event Gateway plug-in databases.
`-plugin` *PluginName*	Name of the plug-in. **Note:** You can run the script for only one plug-in at a time.Plug-in names for use in this command line option are as follows. If the plug-in name consists of more than 1 word, enclose the name in double quotation marks, for example, `"Adaptive Polling"`. • Adaptive Polling • Disco • Failover • Fix Pack 4 PostNcimProcessing • RCA • SAE IP Path • SAE ITNM Service • SAE MPLS VPN • zNetView
`-disable`	Disables the specified plug-in.
`-enable`	Enables the specified plug-in.
`-global`	Enables plug-ins in all domains. If this is not specified, then the plug-in is enabled only in the domain that is specified by the `-domain` option.
`-help`	Displays a full set of help text for the command-line options. **Tip:** For a brief list of available options, run the script with no options.

For example, to enable the zNetView plug-in in all domains, run the script as follows:

© Copyright IBM Corp. 2006, 2016

```
$NCHOME/precision/bin/ncp_perl $NCHOME/precision/scripts/perl/scripts/
ncp_gwplugins.pl -domain NCOMS -plugin zNetView -enable -global
```

Related reference:

"Plugin descriptions" on page 118
Use this information to understand what each Event Gateway plugin does.

"ncp_g_event plug-in database tables in ncmonitor" on page 241
Use this information to understand which Event Gateway configuration tables are available in the ncmonitor database and what type of information each table contains. Most of these tables relate to Event Gateway plug-in configuration.

Listing plug-in information

You can list information on Event Gateway plug-ins. For example, you can list the event maps and event states that each plug-in subscribes to.

Use the `ncp_gwplugins.pl` script to list plug-in information, The script is located at `$NCHOME/precision/scripts/perl/scripts/ncp_gwplugins.pl`.

To run the script to list event map subscriptions, issue a command similar to the following. This example lists all event maps and event states subscribed to by the Disco plug-in.

```
$NCHOME/precision/bin/ncp_perl
$NCHOME/precision/scripts/perl/scripts/ncp_gwplugins.pl -domain NCOMS -plugin Disco
```

Command-line options

The following table describes the command-line options for the `ncp_gwplugins.pl` script used in this example. For help, run the script as follows:

- For a brief list of the available options, run the command without any options.
- For a full set of command line options, run the script with the `-help` option.

Table 49. ncp_gwplugins.pl command-line options

Command-line option	Description
`-domain` *DomainName*	Mandatory; the name of a domain related to the plug-in. This domain is used to enable the script to read the relevant DbLogins,cfg file in order to connect to the relevant Event Gateway plug-in databases.

Table 49. ncp_gwplugins.pl command-line options (continued)

Command-line option	Description
`-plugin` *PluginName*	Name of the plug-in. **Note:** You can run the script for only one plug-in at a time.Plug-in names for use in this command line option are as follows. If the plug-in name consists of more than 1 word, enclose the name in double quotation marks, for example, `"Adaptive Polling"`. • Adaptive Polling • Disco • Failover • Fix Pack 4 PostNcimProcessing • RCA • SAE IP Path • SAE ITNM Service • SAE MPLS VPN • zNetView

Related reference:

"Plugin descriptions" on page 118
Use this information to understand what each Event Gateway plugin does.

"ncp_g_event plug-in database tables in ncmonitor" on page 241
Use this information to understand which Event Gateway configuration tables are available in the ncmonitor database and what type of information each table contains. Most of these tables relate to Event Gateway plug-in configuration.

Modifying event map subscriptions

You can change the event maps that a plug-in subscribes to. For example, if you add a new event map and want the system to perform RCA on events handled by that event map, then you must add that event map to the subscription list for the RCA plug-in.

Use the `ncp_gwplugins.pl` script to modify event map subscriptions. The script is located at `$NCHOME/precision/scripts/perl/scripts/ncp_gwplugins.pl`.

To run the script to modify event map subscriptions, issue a command similar to the following. In this example the event map `PnniIfState` is added to the subscription list for the RCA plug-in.

```
$NCHOME/precision/perl/bin/ncp_perl $NCHOME/precision/scripts/
perl/scripts/ncp_gwplugins.pl -domain NCOMS -plugin RCA -add -eventMap PnniIfState
```

Command-line options

The following table describes the command-line options for the `ncp_gwplugins.pl` script used in this example. For help, run the script as follows:

- For a brief list of the available options, run the command without any options.
- For a full set of command line options, run the script with the `-help` option.

Table 50. ncp_gwplugins.pl command-line options

Command-line option	Description
`-domain` *DomainName*	Mandatory; the name of a domain related to the plug-in. This domain is used to enable the script to read the relevant DbLogins,cfg file in order to connect to and update the relevant Event Gateway plug-in databases.
`-plugin` *PluginName*	Name of the plug-in. **Note:** You can run the script for only one plug-in at a time.Plug-in names for use in this command line option are as follows. If the plug-in name consists of more than 1 word, enclose the name in double quotation marks, for example, `"Adaptive Polling"`. • Adaptive Polling • Disco • Failover • Fix Pack 4 PostNcimProcessing • RCA • SAE IP Path • SAE ITNM Service • SAE MPLS VPN • zNetView
`-add`	For the specified plug-in or plug-ins, adds interest in the specified event map. Requires options `-plugin` and `-eventMap` to be specified.
`-drop`	For the specified plug-in or plug-ins, removes interest in the specified event map.
`-eventMap` *EventMapName*	Event map for which interest is to be added or deleted.

Related concepts:

"Quick reference for event enrichment" on page 83
Use this information to understand how an event is processed as it passes through the Event Gateway.

"Quick reference for RCA" on page 131
Use this information to understand how an event is processed as it passes through the RCA plug-in.

"Outgoing Event Gateway queue" on page 91
The outgoing Event Gateway queue receives enriched events from the Event Gateway stitchers (main event enrichment) and from the plug-ins. In order to minimize the number of updates and hence minimize the load on the ObjectServer, updates to the Object Server are placed in a queue, aggregated, and sent to the ObjectServer at a specified interval. The default is 5 seconds.

Related reference:

"Plugin descriptions" on page 118
Use this information to understand what each Event Gateway plugin does.

"ncp_g_event plug-in database tables in ncmonitor" on page 241
Use this information to understand which Event Gateway configuration tables are available in the ncmonitor database and what type of information each table contains. Most of these tables relate to Event Gateway plug-in configuration.

Setting plug-in configuration parameters

You can set optional configuration parameters for the Event Gateway plug-ins using the `ncp_gwplugins.pl` script.

Use the `ncp_gwplugins.pl` script to set optional configuration parameters. The script is at `$NCHOME/precision/scripts/perl/scripts/ncp_gwplugins.pl`.

To run the script to set configuration parameters, issue a command similar to the following. This example sets the update interval for the ncmonitor.activeEvent table to 10 seconds. The default is 5 seconds.

```
$NCHOME/precision/perl/bin/ncp_perl
$NCHOME/precision/scripts/perl/scripts/ncp_gwplugins.pl -domain NCOMS
-plugin "Adaptive Polling" -set -name ActiveEventUpdateInterval -value 10
```

Command-line options

The following table describes the command-line options for the `ncp_gwplugins.pl` script used in this example. For help, run the script as follows:

- For a brief list of the available options, run the command without any options.
- For a full set of command line options, run the script with the `-help` option.

Table 51. `ncp_gwplugins.pl` command-line options

Command-line option	Description
`-domain` *DomainName*	Required: The name of the domain that is related to the plug-in. The domain tells the script which `DbLogins.`*`domain`*`.cfg` to read so that it can connect to and update the correct Event Gateway plug-in databases.
`-plugin` *PluginName*	Name of the plug-in. **Note:** You can run the script for only one plug-in at a time.Plug-in names for use in this command line option are as follows. If the plug-in name consists of more than 1 word, enclose the name in double quotation marks, for example, `"Adaptive Polling"`. • Adaptive Polling • Disco • Failover • Fix Pack 4 PostNcimProcessing • RCA • SAE IP Path • SAE ITNM Service • SAE MPLS VPN • zNetView
`-set`	Indicates that a variable is to be set.
`-name` *ParameterName*	Name of the parameter to set.
`-value` *Parametervalue*	Value to set for this parameter.

Related reference:

"Plugin descriptions" on page 118
Use this information to understand what each Event Gateway plugin does.

"ncp_g_event plug-in database tables in ncmonitor" on page 241
Use this information to understand which Event Gateway configuration tables are available in the ncmonitor database and what type of information each table contains. Most of these tables relate to Event Gateway plug-in configuration.

Configuring the SAE plug-in

Use this information to understand how to configure the SAE plug-in.

Related concepts:

"SAE plug-in" on page 125
The SAE plug-in generates service-affected events for MPLS VPNs and IP paths.

Configuring summary field information in service-affected events

To make service-affected events more meaningful for operators, you can configure the SAE plug-in to insert customer-related information into the Summary field of a service-affected event.

The configuration files in the SAE plug-in where you make this change are as follows:

- SaeIpPath.cfg for the IP Path service, located at `$NCHOME/etc/precision/SaeIpPath.cfg`
- SaeMplsVpn.cfg for the MPLS VPN service, located at `$NCHOME/etc/precision/SaeMplsVpn.cfg`
- SaeItnmService.cfg for custom services, located at `$NCHOME/etc/precision/SaeItnmService.cfg`

The field used in each of these files to configure extra information to insert into the SAE Summary field is called CustomerNameField. The following example shows how to configure this field in the SaeMplsVpn.cfg file.

1. Open the SaeMplsVpn.cfg configuration file.
2. Modify the insert statement by adding the text in bold to insert data from a relevant field in the service record in NCIM cache into the CustomerNameField field. For example, the following statement will insert the content of the entityData->DESCRIPTION field (if this field exists) into the CustomerNameField, and into the Summary field of any MPLS VPN edge service SAE generated.

   Note: When you add a field to the insert, you must add a comma to the preceding line.

```
insert into config.serviceTypes
(
    ServiceTypeName,
    CollectionEntityType,
    ConstraintFilter,
    CustomerNameField
)
values
(
    "MPLSVPNEdgeService",
    17 -- "networkVpn",
    "networkVpn->VPNTYPE <> 'MPLS Core'",
    "entityData->DESCRIPTION"
```

Adding SAE types to the SAE plug-in

You can configure the SAE plug-in to generate more SAE types than the three provided by default. For example, you can configure the plug-in to create SAE events for MPLS VPN edge entities (one type of SAE) and for MPLS VPN core entities (another type of SAE).

In this example the existing configuration file SaeMplsVpn.cfg is customized to add an extra MPLS VPN SAE service types to the config.serviceTypes table. The new service type is called MPLS VPN Core Service, and generates SAEs when a Severity 5 (critical) fault event occurs on any router in the core network. You can also create new SAE service types by creating a brand new configuration file and specifying the relevant inserts there.

The configuration file for the MPLS VPN SAE service types in the SAE plug-in is the SAEMplsVpn.cfg configuration file. This file is located at: `$NCHOME/etc/precision/SAEMplsVpn.cfg`.

1. Open the SAEMplsVpn.cfg configuration file.
2. The default insert creates an MPLS VPN Edge Service and reads as follows:

```
insert into config.serviceTypes
(
    ServiceTypeName,
    CollectionEntityType,
    ConstraintFilter
)
values
(
    "MPLS VPN Edge Service",
    17,   -- networkVpn
    "networkVpn->VPNTYPE <> 'MPLS Core'"
);
```

3. Add a new insert after the existing insert. The new insert should read as follows:

```
insert into config.serviceTypes
(
    ServiceTypeName,
    CollectionEntityType,
    ConstraintFilter
)
values
(
    "MPLS VPN Core Service",
    17,   -- networkVpn
    "networkVpn->VPNTYPE = 'MPLS Core'"
);
```

Note: You can have two or more SAE service types for a given table such as networkVpn (17), as described in this example. In this case, the SAE service types must be mutually exclusive sets, otherwise one will win over the other where they overlap. For example, the service types described in this example do not overlap because they have complementary `ConstraintFilter` settings as follows:

- `networkVpn->VPNTYPE <> 'MPLS Core'`
- `networkVpn->VPNTYPE = 'MPLS Core'`

Related concepts:

"SAE plug-in" on page 125
The SAE plug-in generates service-affected events for MPLS VPNs and IP paths.

Chapter 14. Configuring root-cause analysis

You can configure the RCA plug-in.

Related reference:

"Resynchronizing events with the ObjectServer" on page 158
Issue the SIGHUP command to the Event Gateway to change the configuration of the Event Gateway.

Configuring the poller entity

To enable the RCA plugin to perform isolated suppression when the Network Manager server is not within the scope of your network domain, specify the IP address or DNS name of the ingress interface as the poller entity.

The configuration file for the Event Gateway is the EventGatewaySchema.cfg configuration file. This file is located at: `$NCHOME/etc/precision/EventGatewaySchema.cfg`. The poller entity value is stored in the config.defaults table, in the field NcpServerEntity.

1. Open the EventGatewaySchema.cfg configuration file.
2. Identify the insert statement into the config.defaults table. By default this insert statement has the following form:

```
insert into config.defaults
(
    IDUCFlushTime,
    ObjectServerUpdateInterval,
    NcpServerEntity
)
values
(
    5,
    5,
    ""
);
```

   By default the NcpServerEntity field is empty. In this case, the Event Gateway searches the topology using the IP address or the addresses of the local host it is running on.
3. Modify this statement to set the NcpServerEntity field to the value of the IP address or DNS name of the ingress interface.

Related concepts:

"Poller entity" on page 133
Use this information to understand what the poller entity is and how to configure it.

Related reference:

Fix Pack 4 "RCA considerations in a cross-domain network" on page 170
In a cross-domain environment, the **ncp_g_event** process in each discovery domain performs RCA on the devices in the same discovery domain. Within each domain, RCA operates in the same way as it does when there is only a single domain. Root Cause can also be analyzed across multiple domains when they are visualized together using a cross-domain discovery.

"config.defaults table" on page 232
The config.defaults table contains general configuration data for the Event

© Copyright IBM Corp. 2006, 2016

Gateway.

Configuring the maximum age difference for events

By default, events on the same entity suppress each other regardless of the age of the events. An event received today can suppress an event received yesterday on the same entity. You can change this by specifying a maximum age difference between events that pass through the RCA plug-in. Events that have a difference in age greater than this specified value cannot suppress each other.

The configuration file for the RCA plug-in is the RCASchema.cfg configuration file. This file is located at: `$NCHOME/etc/precision/RCASchema.cfg`. The value for maximum age difference between events is stored in the config.defaults table, in the field MaxAgeDifference.

1. Open the `RCASchema.cfg` configuration file.
2. Identify the insert statement into the config.defaults table. By default this insert statement has the following form:

```
insert into config.defaults
(
  RequeueableEventIds,
  MaxAgeDifference
  HonourManagedStatus
)
values
(
    [
        'NmosPingFail',
        'NmosSnmpPollFail'
    ],
    0
);
```

   By default the value for MaxAgeDifference is 0. This means that the feature is turned off.
3. Modify this statement to set the MaxAgeDifference field to a desired value in minutes.

   Tip: For example, set the MaxAgeDifference field to a value of 15 to configure the system so that events on the same entity that have a difference in age greater than fifteen minutes cannot suppress each other.

Related reference:

"config.defaults database table" on page 239
The config.defaults database table stores configuration data for the RCA plug-in event queue.

RCA considerations in a cross-domain network

Fix Pack 4

In a cross-domain environment, the **ncp_g_event** process in each discovery domain performs RCA on the devices in the same discovery domain. Within each domain, RCA operates in the same way as it does when there is only a single domain. Root Cause can also be analyzed across multiple domains when they are visualized together using a cross-domain discovery.

Contained suppression

Because the interfaces are in the same domain as the chassis that contains them, contained RCA is unaffected by a cross-domain environment.

Connected interfaces

In a cross-domain environment, most connections are between two interfaces in the same domain and connected suppression works as expected. If the interfaces are in different domains, connected suppression does not associate the events at each end of the link.

Isolated suppression

Devices on the edge of the network, which have fewer connections, can become unreachable (isolated) from the poller entity if a device between them and the poller entity fails. Events on these isolated devices are suppressed, as long as all isolated devices are in the same domain. When you partition your network, ensure that groups of devices that are isolated are kept within the same domain.

SAE RCA

If a service, for example a VPN, consists of devices in two domains, then two SAE events can be created for the same VPN, one in each domain.

Appendix A. Default poll policies

Network Manager IP Edition provides a set of default poll policies. Use this information to familiarize yourself with these policies.

Default ping policies

Network Manager IP Edition provides default poll policies for ping operations.

The following table provides information on the default ping poll policies.

Table 52. Default ping poll policies

Poll Policy Name	Description
Default Chassis Ping	Uses the Default Chassis Ping poll definition to ping all network devices. The type of device pinged is restricted by the Class Filter in the Default Chassis Ping poll definition. This is the only poll policy to be enabled by default.
Default Interface Ping	Uses the Default Interface Ping poll definition to perform ping operations on all interfaces that are within a main node with a valid IP address. This policy uses the Default Interface Ping poll definition to ping interfaces on all network devices. The interfaces pinged are restricted according to the following: • The interface filter in the poll definition. This can be further refined by adding extra poll definition filters. • The managed status of each interface. This can be changed by modifying the TagManagedEntities stitcher.
End Node Ping	Uses the End Node Ping poll definition to perform ping operations on all end nodes, as defined by the class settings.
ConfirmDeviceDown	Uses the Default Chassis Ping poll definition with an increased polling frequency and a policy scope that includes only those devices that on which an NmosPingFail event has occurred. This policy is used as part of an adaptive polling scenario and has the aim of accelerating ping polling of devices that failed to respond to a ping poll in order to identify devices that are really down.

Default remote ping policies

Network Manager IP Edition provides default poll policies for remote ping operations. These polls use SNMP write operations to control vendor-specific extensions to the DISMAN-PING-MIB.

The following table provides information on the default remote ping poll policies.

© Copyright IBM Corp. 2006, 2016

Table 53. Default remote ping poll policies

Poll policy name	Description
Cisco Remote Ping	Uses the Cisco Remote Ping poll definition to check the availability of MPLS paths between Cisco Provider Edge (PE) and Customer Edge (CE) devices through SNMP remote ping operations. This poll policy is applicable only to Cisco devices.
Juniper Remote Ping	Uses the Juniper Remote Ping poll definition to check the availability of MPLS paths between Juniper PE and CE devices through SNMP remote ping operations, as defined by the class settings. This poll policy is applicable only to Juniper devices.

Restriction: Storage of polled data is not supported for the Cisco Remote Ping, the Juniper Remote Ping, and the Generic Threshold poll definitions.

Default SNMP threshold policies

Default SNMP threshold poll policies are provided with the product. These poll policies are classified as *basic* or *generic*; in addition, some are vendor-specific.

Threshold policies

The following table describes the SNMP threshold poll policies.

Table 54. Basic SNMP threshold poll policies

Name	Description
bgpPeerState	Poll definition type: Generic threshold. Uses the bgpPeerState poll definition to perform threshold polling on all network devices with an IP forwarding capability, as defined by the class settings and the scope settings.
ConfirmHighDiscardRate	Poll definition type: Basic threshold. Provides accelerated SNMP polling. Uses the HighDiscardRate poll definition to perform threshold polling on all network devices that have at least one interface that breached the 5% packet discard rate threshold, and as defined by the class settings. This policy is used as part of an adaptive polling scenario and has the aim of accelerating polling to confirm threshold violations on device interfaces.
dot3StatsAlignmentErrors	Poll definition type: Basic threshold. Uses the dot3StatsAlignmentErrors poll definition to perform threshold polling on all network devices, as defined by the class settings.
frCircuitReceivedBECNs	Poll definition type: Basic threshold. Uses the frCircuitReceivedBECNs poll definition to perform threshold polling on all network devices, as defined by the class settings.

Table 54. Basic SNMP threshold poll policies (continued)

Name	Description
frCircuitReceivedFECNs	Poll definition type: Basic threshold. Uses the frCircuitReceivedFECNs poll definition to perform threshold polling on all network devices, as defined by the class settings.
HighDiscardRate	Poll definition type: Basic threshold. Uses the HighDiscardRate poll definition to perform threshold polling on all network devices, as defined by the class settings. The policy polls for this information every 30 minutes.
ifInDiscards	Poll definition type: Basic threshold. Uses the ifInDiscards poll definition to perform threshold polling on all network devices, as defined by the class settings.
ifInErrors	Poll definition type: Basic threshold. Uses the ifInErrors poll definition to perform threshold polling on all network devices, as defined by the class settings.
ifOutDiscards	Poll definition type: Basic threshold. Uses the ifOutDiscards poll definition to perform threshold polling on all network devices, as defined by the class settings.
ifOutErrors	Poll definition type: Basic threshold. Uses the ifOutErrors poll definition to perform threshold polling on all network devices, as defined by the class settings.
frCircuitState	Poll definition type: Generic threshold. Uses the frCircuitState poll definition to perform threshold polling on all network devices, as defined by the class settings.
isdnLinkUp	Poll definition type: Generic threshold. Uses the isdnLinkUp poll definition to perform threshold polling on all network devices, as defined by the class settings.
Fix Pack 5 ospfNbrState	Fix Pack 5 Poll definition type: Generic threshold. Uses the ospfNbrState poll definition to perform threshold polling on OSPF routers.
rebootDetection	Poll definition type: Generic threshold. Uses the rebootDetection poll definition to perform threshold polling on all network devices, as defined by the class settings.
snmpInBandwidth	Poll definition type: Basic threshold. Uses the snmpInBandwidth poll definition to perform threshold polling on all network devices, as defined by the class settings.
snmpOutBandwidth	Poll definition type: Basic threshold. Uses the snmpOutBandwidth poll definition to perform threshold polling on all network devices, as defined by the class settings.

Foundry SNMP threshold poll policies

The following table describes the SNMP threshold policies that are supplied for Foundry devices.

Table 55. SNMP threshold poll policies for Foundry devices

Name	Description
snChasActualTemperature	Uses the snChasActualTemperature poll definition to perform threshold polling on all Foundry devices, as defined by the class settings.
snChasFanOperStatus	Uses the snChasFanOperStatus poll definition to perform threshold polling on all Foundry devices, as defined by the class settings.
snChasPwrSupplyOperStatus	Uses the snChasPwrSupplyOperStatus poll definition to perform threshold polling on all Foundry devices, as defined by the class settings.

Cisco SNMP threshold policies

The following table describes the SNMP threshold poll policies that are provided for Cisco devices.

Table 56. SNMP threshold poll policies for Cisco devices

Name	Description
bufferPoll	Uses the bufferPoll poll definition to perform threshold polling on all Cisco devices, as defined by the class settings.
ciscoEnvMonFanState	Uses the ciscoEnvMonFanState poll definition to perform threshold polling on all Cisco devices, as defined by the class settings.
ciscoEnvMonSupplyState	Uses the ciscoEnvMonSupplyState poll definition to perform threshold polling on all Cisco devices, as defined by the class settings.
ciscoEnvMonTemperature State	Uses the ciscoEnvMonTemperatureState poll definition to perform threshold polling on all Cisco devices, as defined by the class settings.
memoryPoll	Uses the ciscoMemoryPool poll definition to perform threshold polling on all Cisco devices, as defined by the class settings.
cpuBusyPoll	Uses the cpuBusyPoll poll definition to perform threshold polling on all Cisco devices, as defined by the class settings.
locIfInCrcErrors	Uses the locIfInCrcErrors poll definition to perform threshold polling on all Cisco devices, as defined by the class settings.
memoryPoll	Uses the memoryPoll poll definition to perform threshold polling on all Cisco devices, as defined by the class settings.
sysTrafficPoll	Uses the sysTrafficPoll poll definition to perform threshold polling on all Cisco devices, as defined by the class settings.

Default SNMP link state policies

The default SNMP Link State poll policy uses the `SNMP Link State` poll definition to check the administrative and operational statuses of all network devices, as defined by the class settings. Events are generated if there are changes in interface status.

Poll policies used by reporting

There are no poll policies defined specifically for reporting. Use this information to understand which existing poll policies are used by reports.

Certain reports require specific poll policies to be enabled. These reports and policies are defined in Table 57.

Table 57. Poll policies used by reporting

Report	Poll policy that must be enabled
Device Egress Traffic Summary	ifOutError ifOutDiscards snmpOutBandwidth
Device Ingress Traffic Summary -	ifInError ifInDiscards snmpInBandwidth
Router Health Summary -	ciscoCPUTotal5Min ciscoMemoryPctgUsage Default Chassis Poll

Appendix B. Default poll definitions

Network Manager IP Edition provides a number of default poll definitions that fulfil the most common polling requirements.

The following table describes the default poll definitions that Network Manager IP Edition provides.

Default SNMP poll definitions

bgpPeerState

Poll definition type: Generic threshold. This poll definition defines BGP peer-state checking. An alert is raised when Border Gateway Patrol (BGP) peers change to an unestablished state. This poll definition polls the bgpPeerState MIB variable, which has the following path and OID:

Path: `iso/org/dod/mgmt/mib-2/bgpPeerTable/bgpPeerEntry/bgpPeerState`

MIB OID: 1.3.6.1.2.1.15.3.1.2

bufferPoll

Poll definition type: Basic threshold. This poll definition defines buffer-exhaustion checking. An alert is raised when the number of free buffer elements falls below 100. This poll definition polls the bufferElFree MIB variable, which has the following path and OID:

MIB path: `iso/org/dod/private/enterprises/cisco/local/bufferElFree`

MIB OID: 1.3.6.1.4.1.9.2.1.9

ciscoCPUTotal5min

Poll definition type: Basic threshold. This poll definition defines CPU usage checking. An alert is raised when the value of the cpmCPUTotal5min Cisco MIB variable exceeds 80%. This poll definition polls the cpmCPUTotal5min MIB variable, which shows the overall CPU busy percentage in the last five-minute period. This object deprecates the avgBusy5 object from the OLD-CISCO-SYSTEM-MIB. The cpmCPUTotal5min MIB variable has the following path and OID:

MIB path: `iso/org/dod/private/enterprises/cisco/local/cpmCPUTotal5min`

MIB OID: 1.3.6.1.4.1.9.9.109.1.1.1.1.5

ciscoEnvMonFanState

Poll definition type: Generic threshold. This poll definition defines fan-status checking for Cisco devices. An alert is raised if the status changes to anything other than 1 (normal). This poll definition polls the ciscoEnvMonFanState MIB variable, which has the following path and OID:

MIB Path: `iso/org/dod/mgmt/mib-2/private/enterprises/cisco/ciscoMgmt/ciscoEnvMonMIB/ciscoEnvMonObjects/ciscoEnvMonFanStatusTable/ciscoEnvMonFanStatusEntry/ciscoEnvMonFanState`

MIB OID: 1.3.6.1.4.1.9.9.13.1.4.1.3

© Copyright IBM Corp. 2006, 2016

ciscoEnvMonSupplyState

Poll definition type: Generic threshold. This poll definition defines the checking of the power-supply status for Ciso devices. An alert is raised if the status changes to anything other than 1 (normal). This poll definition polls the ciscoEnvMonSupplyState MIB variable, which has the following path and OID:

MIB path: `iso/org/dod/mgmt/mib-2/private/enterprises/cisco/ciscoMgmt/ciscoEnvMonMIB/ciscoEnvMonObjects/ciscoEnvMonSupplyStatusTable/ciscoEnvMonSupplyStatusEntry/ciscoEnvMonSupplyState`

MIB OID: 1.3.6.1.4.1.9.9.13.1.5.1.3

ciscoEnvMonTemperatureState

Poll definition type: Generic threshold. This poll definition defines the checking of the fan-temperature status for Cisco devices. An alert is raised if the status changes to anything other than 1 (normal). This poll definition polls the ciscoEnvMonTemperatureState MIB variable, which has the following path and OID:

MIB path: `iso/org/dod/mgmt/mib-2/private/enterprises/cisco/ciscoMgmt/ciscoEnvMonMIB/ciscoEnvMonObjects/ciscoEnvMonTemperatureStatusTable/ciscoEnvMonTemperateureStatusEntry/ciscoEnvMonTemperatureState`

MIB OID: 1.3.6.1.4.1.9.9.13.1.3.1.6

Cisco Remote Ping

Poll definition type: Cisco remote ping. This poll definition defines remote ping operations that use Cisco-specific MIBs.

cpuBusyPoll

Poll definition type: Basic threshold. This poll definition defines CPU usage checking. An alert is raised when the value of the avgBusy5 Cisco MIB variable exceeds 80%. This poll definition polls the avgBusy5 MIB variable, which has the following path and OID:

MIB path: `iso/org/dod/private/enterprises/cisco/local/avgBusy5`

MIB OID: 1.3.6.1.4.1.9.2.1.58

Restriction: The aveBusy5 MIB variable is not supported on some recent Cisco routers. For such routers, use the ciscoCPUTotal5min poll definition.

HighDiscardRate

Poll definition type: Basic threshold. An alert is raised when the packet discard rate on at least one interface on a device exceeds 5%. This poll definition polls the following MIB variables:

ifInDiscards

MIB path: `iso/org/dod/mgmt/mib-2/interfaces/ifTable/ifEntry/ifInDiscards`

MIB OID: 1.3.6.1.2.1.2.2.1.13

ifInNUcastPkts

MIB path: `iso/org/dod/mgmt/mib-2/interfaces/ifTable/ifEntry/ifInNUcastPkts`

MIB OID: 1.3.6.1.2.1.2.2.1.12

ifInUcastPkts

MIB path: `iso/org/dod/mgmt/mib-2/interfaces/ifTable/ifEntry/ifInUcastPkts`

MIB OID: 1.3.6.1.2.1.2.2.1.11

dot3StatsAlignmentErrors

Poll definition type: Basic threshold. This poll definition defines the checking of error rates for alignments. An alert is raised when the error rate exceeds 0 per second. This poll definition polls the dot3StatsAlignmentErrors MIB variable, which has the following path and OID:

MIB path: `iso/org/dod/mgmt/mib-2/transmission/dot3/dot3StatsTable/dot3StatusEntry/dot3StatsAlignmentErrors`

MIB OID: 1.3.6.1.2.1.10.7.2.1.2

frCircuitReceivedBECNs

Poll definition type: Basic threshold. This poll definition defines Frame Relay backward-congestion checking for a data link connection identifier (DLCI). An alert is raised when backward-congestion notices are received for DLCI. This poll definition polls the frCircuitReceivedBECNs MIB variable, which has the following path and OID:

MIB path: `iso/org/dod/mgmt/mib-2/transmission/frameRelayDTE/frCircuitTable/frCircuitEntry/frCircuitReceivedBECNs`

MIB OID: 1.3.6.1.2.1.10.32.2.1.5

frCircuitReceivedFECNs

Poll definition type: Basic threshold. This poll definition defines Frame Relay forward-congestion checking for DLCI. An alert is raised when forward-congestion notices are received for DLCI. This poll definition polls the frCircuitReceivedFECNs MIB variable, which has the following path and OID:

MIB path: `iso/org/dod/mgmt/mib-2/transmission/frameRelayDTE/frCircuitTable/frCircuitEntry/frCircuitReceivedFECNs`

MIB OID: 1.3.6.1.2.1.10.32.2.1.4

frCircuitState

Poll definition type: Generic threshold. This poll definition defines Frame Relay circuit-state checking. An alert is raised when a circuit becomes inactive. To avoid generating alerts for circuits that are inactive at startup, the definition checks for inactive circuits. This poll definition polls the frCircuitState MIB variable, which has the following path and OID:

MIB path: `iso/org/dod/mgmt/mib-2/transmission/frameRelayDTE/frCircuitTable/frCircuitEntry/frCircuitState`

MIB OID: 1.3.6.1.2.1.10.32.2.1.3

ifInDiscards

Poll definition type: Basic threshold. This poll definition defines inbound discard-rate checking. An alert is raised when the rate exceeds 0 per second. This poll definition polls the ifInDiscards MIB variable, which has the following path and OID:

MIB path: `iso/org/dod/mgmt/mib-2/interfaces/ifTable/ifEntry/ifInDiscards`

MIB OID: 1.3.6.1.2.1.2.2.1.13

ifInErrors

Poll definition type: Basic threshold. This poll definition defines inbound interface error-rate checking. An alert is raised when the rate exceeds 0 per second. This poll definition polls the ifInErrors MIB variable, which has the following path and OID:

MIB path:iso/org/dod/mgmt/mib-2/interfaces/ifTable/ifEntry/ifInErrors

MIB OID: 1.3.6.1.2.1.2.2.1.14

ifOutDiscards

Poll definition type: Basic threshold. This poll definition defines outbound discard-rate checking. An alert is raised when the rate exceeds 0 per second. This poll definition polls the ifOutDiscards MIB variable, which has the following path and OID:

MIB path: iso/org/dod/mgmt/mib-2/interfaces/ifTable/ifEntry/ifOutDiscards

MIB OID: 1.3.6.1.2.1.2.2.1.19

ifOutErrors

Poll definition type: Basic threshold. This poll definition defines the rate of error checking for outbound interfaces. An alert is raised when the rate exceeds 0 per second. This poll definition polls the ifOutErrors MIB variable, which has the following path and OID:

MIB path:iso/org/dod/mgmt/mib-2/interfaces/ifTable/ifEntry/ifOutErrors

MIB OID: 1.3.6.1.2.1.2.2.1.20

isdnLinkUp

Poll definition type: Generic threshold. This poll definition defines ISDN link-state checking. An alert is raised when an ISDN link is activated. The activation of an ISDN link might indicate that a corresponding primary link has gone down. This poll definition polls the following MIB variables:

MIB path: iso/org/dod/mgmt/mib-2/interfaces/ifTable/ifEntry/ifOperStatus

MIB OID: 1.3.6.1.2.1.2.2.1.8

MIB path: iso/org/dod/mgmt/mib-2/interfaces/ifTable/ifEntry/ifAdminStatus

MIB OID: 1.3.6.1.2.1.2.2.1.7

Juniper Remote Ping

Poll definition type: Juniper remote ping. This poll definition defines remote ping operations that use Juniper-specific MIBs.

locIfInCrcErrors

Poll definition type: Basic threshold. This poll definition defines inbound cyclic redundancy checksum (CRC) /alignment error checking. An alert is raised when inbound CRC alignment errors occur. This poll definition polls the locIfInCRC MIB variable, which has the following path and OID:

MIB path: iso/org/dod/mgmt/mib-2/private/enterprises/cisco/local/linterfaces/lifTable/lifEntry/locIfInCRC

MIB OID: 1.3.6.1.4.1.9.2.2.1.1.12

memoryPoll

Poll definition type: Basic threshold. This poll definition defines memory-exhaustion checking. An alert is raised when the amount of free memory falls below 100 bytes. This poll definition polls the freeMem MIB variable, which has the following path and OID:

MIB path: iso/org/dod/mgmt/mib-2/private/enterprises/cisco/local/lsystem/freeMem

MIB OID: 1.3.6.1.4.1.9.2.1.8

Fix Pack 5 **ospfNbrState**

Poll definition type: Generic threshold. This poll definition defines OSPF adjacency checking, and generates an alert if an OSPF router is in a state other than `full`. If an OSPF router is in a state of `down`, `attempt`, `init`, `twoWay`, `exchangeStart`, `exchange`, or `loading`, it indicates a problem in forming adjacencies. This poll definition polls the `ospfNbrState` MIB variable, which has the following path and OID:

MIB path: `iso/org/dod/internet/mgmt/mib-2/ospf/ospfNbrTable/ospfNbrEntry/ospfNbrState`

MIB OID: 1.3.6.1.2.1.14.10.1.6

rebootDetection

Poll definition type: Generic threshold. This poll definition defines device-reboot detection checking where an alert is generated if a device is rebooted. The reason for a device reboot might be that the SNMP subsystem on a device has been reset.

Tip: To monitor system uptime, change the sysUpTime MIB variable to the hrSystemUptime MIB variable. The hrSystemUptime MIB variable is available only if the HOSTRES-MIB is supported by the device.
This poll definition polls the sysUpTime MIB variable, which has the following path and OID:

MIB path: `iso/org/dod/mgmt/mib-2/system/sysUpTime`

MIB OID: 1.3.6.1.2.1.1.3

snChasActualTemperature

Poll definition type: Generic threshold. This poll definition defines temperature checking for Foundry devices. An alert is raised when the actual temperature of the chassis exceeds the value set for a warning about the temperature level. This poll definition polls the following MIB variables:

snChasActualTemperature

MIB path: `iso/org/dod/mgmt/mib-2/private/enterprises/foundry/foundryProducts/switch/snChassis/snChassGen/snChasActualTemperature`

MIB OID: 1.3.6.1.4.1.1991.1.1.1.1.18

snChasWarningTemperature

MIB path: `iso/org/dod/mgmt/mib-2/private/enterprises/foundry/foundryProducts/switch/snChassis/snChassGen/snChasWarningTemperature`

MIB OID: 1.3.6.1.4.1.1991.1.1.1.1.19

snChasFanOperStatus

Poll definition type: Generic threshold. This poll definition defines fan-status checking for Foundry devices. An alert is raised when the fan status changes from 2 (normal) to 3 (failure). This poll definition polls the snChasFanOperStatus MIB variable, which has the following path and OID:

MIB path: `iso/org/dod/mgmt/mib-2/private/enterprises/foundry/foundryProducts/switch/snChassis/snChasFan/snChasFanTable/snChasFanEntry/snChasFanOperStatus`

MIB OID: 1.3.6.1.4.1.1991.1.1.1.3.1.1.3

snChasPwrSupplyOperStatus

Poll definition type: Generic threshold. This poll definition defines the checking of the power-supply status for Foundry devices. An alert is raised when the power supply status changes from 2 (normal) to 3 (failure). This poll definition polls the snChasPwrSupplyOperStatus MIB variable, which has the following path and OID:

MIB path:`iso/org/dod/mgmt/mib-2/private/enterprises/foundry/foundryProducts/switch/snChassis/snChasPwr/snChasPwrSupplyTable/snChasPwrSupplyEntry/snChasPwrSupplyOperStatus`

MIB OID: 1.3.6.1.4.1.1991.1.1.1.2.1.1.3

SNMP Link State

Poll definition type: SNMP link state. This poll definition defines administrative and operational status checking. An alert is raised if a mismatch occurs between the administrative and operational statuses. This poll definition polls the following MIB variables:

ifAdminStatus

MIB path: `iso/org/dod/mgmt/mib-2/interfaces/ifTable/ifEntry/ifAdminStatus`

MIB OID: 1.3.6.1.2.1.2.2.1.7

ifOperStatus

MIB path: `iso/org/dod/mgmt/mib-2/interfaces/ifTable/ifEntry/ifOperStatus`

MIB OID: 1.3.6.1.2.1.2.2.1.8

snmpInBandwidth

Poll definition type: Basic threshold. This poll definition defines the checking of incoming bandwidth utilization. An alert is raised when incoming bandwidth usage exceeds 40%. This poll definition polls the following MIB variables:

ifInOctets

MIB path: `iso/org/dod/mgmt/mib-2/interfaces/ifTable/ifEntry/ifInOctets`

MIB OID: 1.3.6.1.2.1.2.2.1.10

ifSpeed

MIB path: `iso/org/dod/mgmt/mib-2/interfaces/ifTable/ifEntry/ifSpeed`

MIB OID: 1.3.6.1.2.1.2.2.1.5

snmpOutBandwidth

Poll definition type: Basic threshold. This poll definition defines the checking of outgoing SNMP bandwidth utilization. An alert is raised when the outgoing SNMP bandwidth utilization is above 40% for an interface. This poll definition polls the following MIB variables:

ifOutOctets

MIB path: `iso/org/dod/mgmt/mib-2/interfaces/ifTable/ifEntry/ifOutOctets`

MIB OID: 1.3.6.1.2.1.2.2.1.16

ifSpeed

MIB path: `iso/org/dod/mgmt/mib-2/interfaces/ifTable/ifEntry/ifSpeed`

MIB OID: 1.3.6.1.2.1.2.2.1.5

sysUpTime

Poll definition type: Basic threshold. This poll retrieves the value of the sysUpTime MIB variable, which has the following path and OID:

MIB path: `iso/org/dod/mgmt/mib-2/system/sysUpTime`

MIB OID: 1.3.6.1.2.1.1.3

Default Ping poll definitions

Default Chassis Ping

Poll definition type: Chassis ping. This poll definition defines ping operations for main node devices It sends ICMP packets to the main-node IP address of a device.

Default Interface Ping

Poll definition type: Interface ping. This poll definition defines ping operations for interfaces within devices. It sends ICMP packets to the IP address of each interface.

End Node Ping

Poll definition type: Chassis ping. This poll definition defines ping operations for end nodes, such as printers and workstations. It sends ICMP packets to the IP address of each end node.

For each of these ping poll definitions, the following ping metrics can be collected: response time and packet loss.

Appendix C. Example trigger and clear thresholds

Use the example threshold formulas to set up the clear and trigger thresholds for generic threshold poll definitions.

Example trigger threshold

The following example would raise an event in the following cases:

- When the current value of the avgBusy5 MIB variable is equal to or greater than the value of the avgBusy6 MIB variable

To create this threshold, specify the following information on the **Trigger Threshold** tab of the Poll Definition Editor:

1. Select **Basic**.
2. Specify the first threshold:
   a. Select **Current**.
   b. Click **Add MIB Object** .
   c. Expand the **MIB Tree** to the following path:
      `iso/org/dod/internet/private/enterprises/cisco/local/lsystem/avgBusy5`

      Click **Insert**
   d. Select the comparator `>=`.
   e. Select **current**.
   f. Click **Add MIB Object** .
   g. Expand the MIB Tree to the following path:
      `iso/org/dod/internet/private/enterprises/cisco/local/lsystem/avgBusy6`

      Click **Insert**.
3. Specify the message that is displayed in the Active Event List when the event is raised:
   a. In the **Event description** field, type `CPU usage high (avgBusy5=` .
   b. Click **Add MIB Object** .
   c. Expand the **MIB Tree** to the following path:
      `iso/org/dod/internet/private/enterprises/cisco/local/lsystem/avgBusy5`
   d. Select **Current SNMP Value** and click **Insert**.
   e. Type `>=`.
   f. Click **Add MIB Object** .
   g. Expand the **MIB Tree** to the following path:
      `iso/org/dod/internet/private/enterprises/cisco/local/lsystem/avgBusy6`
   h. Type `)`.

   The description for the Active Event List should now read as follows:

   ```
   CPU usage high (avgBusy5=eval(text,"SNMP.VALUE.avgBusy5")>=eval
   (text,"SNMP.VALUE.avgBusy6"))
   ```

© Copyright IBM Corp. 2006, 2016

Example clear threshold

The following example would raise a Clear event in the following cases:

- When the value of the avgBusy5 MIB variable is less than 80.

To create this threshold, specify the following information on the **Clear Threshold** tab of the Poll Definition Editor:

1. Select **Basic**.
2. Specify the threshold:
   a. Select **Current**.
   b. Click **Add MIB Object** .
   c. Expand the **MIB Tree** to the following path:
      `iso/org/dod/internet/private/enterprises/cisco/local/lsystem/avgBusy5`

      Click **Insert**
   d. Select the comparator `<=`.
   e. Select **literal**.
   f. Type `80`.
3. Specify the message that is displayed in the Active Event List when the event is raised:
   a. In the **Event description** field, type `CPU usage high (avgBusy5=` .
   b. Click **Add MIB Object** .
   c. Expand the **MIB Tree** to the following path:
      `iso/org/dod/internet/private/enterprises/cisco/local/lsystem/avgBusy5`
   d. Select **Current SNMP Value** and click **Insert**.
   e. Type `<=`.
   f. Type `80`.
   g. Type `)`.

   The description for the Active Event List should now read as follows:
   `CPU usage high (avgBusy5=eval(text,"SNMP.VALUE.avgBusy5")<=80)`

Appendix D. Syntax for poll definition expressions

Use this information to understand how to build complex threshold expressions to use in basic and generic threshold poll definitions.

Related reference:

"Example basic threshold expression" on page 44
Use this example basic threshold expression to understand how to compose complex basic threshold expressions.

"Example generic threshold expression" on page 44
Use this example generic threshold expression to understand how to compose complex generic threshold expressions.

eval statement syntax in threshold expressions

Use this information to understand how to use the `eval` statement to create complex threshold expressions within basic and generic threshold poll definitions.

eval statement syntax for SNMP variables

You can evaluate SNMP variables using the `eval` statement.

The following examples illustrate how to evaluate SNMP variables using the `eval` statement.

Sample: Evaluation of SNMP values

The following example returns the value of the SNMP variable `sysName`.

```
eval(text, '&SNMP.VALUE.sysName')
```

Sample: Evaluation of SNMP indices

The following example returns the value of the index of the SNMP request for the variable `ipRouteNextHop`. In a table poll, this is evaluated for every index in the table list..

```
eval(text, '&SNMP.INDEX.ipRouteNextHop')
```

Sample: Evaluation of previously retrieved SNMP values

The following example returns the value of the SNMP variable `sysName`, which was retrieved when this poll was last run..

```
eval(text, '&SNMP.VALUE.OLD.sysName')
```

Sample: Evaluation of Old SNMP Indices

The following example returns the value of the index of the SNMP request for the variable `ipRouteNextHop`, which was retrieved when this poll was last run. In a table poll, this is evaluated for every index in the table list. Note that the old index is likely to be the same as the new index.

```
eval(text, '&SNMP.INDEX.OLD.ipRouteNextHop')
```

© Copyright IBM Corp. 2006, 2016

eval statement syntax for network entity variables

You can evaluate network entity variables, such as the value of an entity ID or entity name, using the `eval` statement.

The `ENTITY` keyword can be used in threshold expressions or descriptions to evaluate the value of network entity variables. The following examples illustrate how to evaluate network entity variables using the `ENTITY` keyword in the `eval` statement.

Sample: Evaluation of the value of the entityName of the containing chassis

The following example can be used in a threshold expression or threshold description, and shows how to evaluate the value of the entityName corresponding to the chassis that contains the entity being monitored.

```
An interface on eval(text, '&ENTITY.MAINNODEENTITYNAME') is down
```

Attributes of the `ENTITY` keyword

Valid attributes of the `ENTITY` keyword are listed in the following table. For information on the NCIM database fields cited in the table, see the *IBM Tivoli Network Manager IP Edition Topology Database Reference*.

Table 58. Attributes of the `ENTITY` keyword

Attribute	Description
`ENTITYID`	Value of the field entity.entityId for the monitored entity. This can be either an interface or chassis.
`ENTITYNAME`	Value of the field entity.entityName for the monitored entity. This can be either an interface or chassis.
`ENTITYTYPE`	Value of the field entity.entityType for the monitored entity. This can be either an interface or chassis.
`ENTITYCLASS`	Value of the field entity.className for the monitored entity. This can be either an interface or chassis.
`ACCESSIPADDRESS`	Value of the field interface.accessIPAddress or chassis.accessIPAddress, depending on what type of poll it is.
`IFINDEX`	For interface polls, the value of the field interface.ifIndex for the monitored entity.
`IFTYPESTRING`	For interface polls, the value of the field interface.ifTypeString for the monitored entity
`IFNAME`	For interface polls, the value of the field interface.ifName for the monitored entity.
`IFDESCR`	For interface polls, the value of the field interface.ifDescr for the monitored entity.
`IFALIAS`	For interface polls, the value of the field interface.ifAlias for the monitored entity.
`INSTANCESTR`	For interface polls, a string representation of the field interface.instanceStr for the monitored entity.
`ENTITYMANAGED`	Indicates whether the monitored entity is in a managed state, determined by whether the field managedStatus.status is either not present or zero for the entity in question.

Table 58. Attributes of the ENTITY keyword (continued)

Attribute	Description
CHASSISMANAGED	Indicates whether the chassis containing the entity being monitored is in a managed status, determined by whether the field managedStatus.status is either not present or zero for the chassis in question.
MAINNODEADDRESS	Value of accessIPAddress for the chassis containing the entity being monitored
MAINNODEENTITYNAME	Value of the field entityName for the entity record corresponding to the chassis containing the entity being monitored.
MAINNODEENTITYID	Value of the field chassis.entityId for the chassis containing the entity being monitored

eval statement syntax for poll policy variables

You can evaluate poll policy variables, such as the name of a policy or the ID of the domain in which this poll policy is found, using the `eval` statement.

The `POLICY` keyword can be used in threshold expressions or descriptions to evaluate the value of poll policy variables. The following examples illustrate how to evaluate poll policy variables using the `POLICY` keyword in the `eval` statement.

Sample: Evaluation of the value of poll policy name and related domain ID

The following example can be used in a threshold expression or threshold description, and shows how to evaluate the value of name of the poll policy and the ID of the domain in which this poll policy is found.

```
The eval(text, '&POLICY.POLICYNAME') policy polls entities in
domain number eval(text, '&POLICY.DOMAINMGRID)
```

Attributes of the POLICY keyword

Valid attributes of the `POLICY` keyword are listed in the following table. For information on the NCIM database fields cited in the table, see the *IBM Tivoli Network Manager IP Edition Topology Database Reference*.

Table 59. Attributes of the POLICY keyword

Attribute	Description
POLICYID	Unique integer identifier for this poll policy.
DOMAINMGRID	Foreign key referencing the NCIM domainMgr table. Specifies the ID for the domain of the monitored entity.
POLICYNAME	Name of this poll policy.

eval statement syntax for poll definition variables

You can evaluate poll definition variables, such as the name of a poll definition or the severity of failure events raised by policies using a poll definition, using the `eval` statement.

The `POLL` keyword can be used in threshold expressions or descriptions to evaluate the value of poll definition variables. The following examples illustrate how to evaluate poll definition variables using the `POLL` keyword in the `eval` statement.

Sample: Evaluation of the value of poll definition name and associated event severity

The following example can be used in a threshold expression or threshold description, and shows how to evaluate the value of name of the poll definition and the severity of the events generated by poll policies that use this poll definition.

```
Poll policies that use the eval(text, '&POLL.TEMPLATENAME') poll definition
generate events with severity eval(text, '&POLL.EVENTSEVERITY')
```

Attributes of the `POLL` keyword

Valid attributes of the `POLL` keyword are listed in the following table.

Table 60. Attributes of the `POLL` keyword

Attribute	Description
`TEMPLATEID`	Unique identifier for this poll definition.
`TEMPLATENAME`	Name of this poll definition.
`TEMPLATETYPE`	Type of poll definition. This value is derived from the list the user is presented with when creating a new poll definition.
`EVENTNAME`	Text identifier to be used for events raised by poll policies that use this poll definition. This text is written to the alerts.status table as the EventId field, unless the text is modified by the rules file of the probe for Tivoli Netcool/OMNIbus (nco_p_ncpmonitor).
`EVENTSEVERITY`	Severity of failure events raised by poll policies using this poll definition. This text is written to the alerts.status table as the Severity field, unless the text is modified by the rules file of the probe for Tivoli Netcool/OMNIbus (nco_p_ncpmonitor).
`POLLINTERVAL`	Interval in seconds at which each entity in scope for this poll is polled.

Operators in threshold expressions

Use this information to understand which operators to use to create threshold expressions in basic and generic threshold poll definitions.

The following table lists the operators that you can use in threshold expressions.

Table 61. Operators in threshold expressions

Operator	Example
Plus	(1 + 2)
Minus	(4 - 2)
Multiplication	(5 * 3)

Table 61. Operators in threshold expressions (continued)

Operator	Example
Division	(10 / 2)
Modulus	(8 % 3)
Power	(10 POW 3)
Log	(Ln 5)
IP to Long datatype conversion	(IpToLong("1.2.3.4"))
Bitwise AND	(5 & 3) **Note:** Bitwise operations can only be applied to integer values.
Bitwise	(5 | 3)
Bitwise Exclusive OR	(5 ^ 3)
Boolean OR	((eval(int, '&SNMP.VALUE.ifSpeed') > 10000) OR (eval(int, '&SNMP.VALUE.ifSpeed') < 100))
Boolean AND	((eval(int, '&SNMP.VALUE.ifSpeed') > 10000) AND (eval(int, '&SNMP.VALUE.ifOperStatus') !=2))
Boolean NOT	(NOT((eval(int,'&SNMP.VALUE.ifOperStatus') = 1))
Equal	(eval(int, '&SNMP.VALUE.ifOperStatus') = 1)
Not equal	(eval(int, '&SNMP.VALUE.ifOperStatus') != 1)
Less than	(eval(int, '&SNMP.VALUE.ifSpeed') < 100)
Greater than	(eval(int, '&SNMP.VALUE.ifSpeed') >100)
Less than or equal	(eval(int, '&SNMP.VALUE.ifSpeed') <= 100)
Greater than or equal	(eval(int, '&SNMP.VALUE.ifSpeed') >= 100)
Like	(eval(text, '&ENTITY.IFDESCR') LIKE 'Gigabit.*')
Not Like	(eval(text, '&ENTITY.IFDESCR') NOT LIKE 'Loopback.*')

Appendix E. Configuration of the Probe for Tivoli Netcool/OMNIbus

The Probe for Tivoli Netcool/OMNIbus (**nco_p_ncpmonitor**) acquires and processes the events that are generated by Network Manager polls and processes, and forwards these events to the ObjectServer.

The Probe for Tivoli Netcool/OMNIbus is installed in the `$NCHOME/probes/`*arch* directory, where *arch* represents an operating system directory. You can configure the probe by using its configuration files, which are as follows:

- Properties file: `nco_p_ncpmonitor.props`
- Rules file: `nco_p_ncpmonitor.rules`

Note: The executable file (or **nco_p_ncpmonitor** command) for the probe is also installed in the `$NCHOME/probes/`*arch* directory. The probe is, however, configured to run under the domain process controller CTRL, by default, and the **nco_p_ncpmonitor** command should be run manually only for troubleshooting purposes.

The events raised in Network Manager are domain-specific. When Network Manager runs in failover mode, the probe uses the virtual domain name by default, provided the name is configured in the `$NCHOME/etc/precision/ConfigItnm.cfg` file.

For more information about probe concepts, see the *IBM Tivoli Netcool/OMNIbus Probe and Gateway Guide* in the Tivoli Netcool/OMNIbus information centre at http://publib.boulder.ibm.com/infocenter/tivihelp/v8r1/topic/com.ibm.tivoli.nam.doc/welcome_ob.htm.

About the nco_p_ncpmonitor.props file

The `$NCHOME/probes/`*arch*`/nco_p_ncpmonitor.props` file defines the environment in which the Probe for Tivoli Netcool/OMNIbus runs.

The properties file is formed of name-value pairs that are separated by a colon. The default properties file lists a subset of the properties that the probe supports; these properties are commented out with a number sign (#) at the beginning of the line. The standard set of common probe properties, which are applicable for the version of Tivoli Netcool/OMNIbus being run, can be specified for the Probe for Tivoli Netcool/OMNIbus, where relevant.

A suggested practice for changing the default values of the properties is that you add a name-value line for each required property at the bottom of the file. To specify a property, ensure that the line is uncommented and then modify the value as required. String values must be enclosed in quotation marks; other values do not require quotation marks. For example:

```
Server          :    "VIRTUAL"
RulesFile       :    "$NCHOME/probes/solaris2/nco_p_ncpmonitor.rules"
Buffering       :    1
BufferSize      :    15
```

© Copyright IBM Corp. 2006, 2016

For troubleshooting purposes, you can alternatively configure probe properties from the command line by running the **nco_p_ncpmonitor** command with the relevant command-line options.

For information about the properties that are common to probes, see the *IBM Tivoli Netcool/OMNIbus Probe and Gateway Guide* in the Tivoli Netcool/OMNIbus information centre at http://publib.boulder.ibm.com/infocenter/tivihelp/v8r1/topic/com.ibm.tivoli.nam.doc/welcome_ob.htm.

About the nco_p_ncpmonitor.rules file

The `$NCHOME/probes/arch/nco_p_ncpmonitor.rules` file defines how the Probe for Tivoli Netcool/OMNIbus should process Network Manager event data to create a meaningful Tivoli Netcool/OMNIbus event.

nco_p_ncpmonitor.rules configuration reference

The rules file maps Network Manager event data to ObjectServer fields, and can be used to customize the behavior of the probe. Knowledge of the Tivoli Netcool/OMNIbus probe rules syntax is required for rules file configuration.

The probe uses tokens and elements, and applies rules, to transform Network Manager event source data into a format that the ObjectServer can recognize. The raw event source data is converted to tokens, which are then parsed into elements. The rules file is used to perform conditional processing on the elements, and to map them to ObjectServer alerts.status fields. In the rules file, elements are identified by the $ symbol and alerts.status fields are identified by the @ symbol. The rules file configuration maps elements to fields, as shown in the following sample code:

```
@Summary=$Description
```

In this example, @Summary identifies the alerts.status field, and $Description identifies the Network Manager input field.

Where the Network Manager ExtraInfo field is used with nested fields to store additional data on entities (for example, ExtraInfo->ifIndex), these fields are available in the following format in the rules file:

```
$ExtraInfo_variable
```

Where *variable* represents a Management Information Base (MIB) variable (for example, ifIndex), or other data (for example, column names in NCIM tables). MIB variables are specified in mixed case characters, and other data, in uppercase characters. For example:

```
$ExtraInfo_ifIndex
$ExtraInfo_MONITOREDENTITYID
```

To configure the rules file for the Probe for Tivoli Netcool/OMNIbus, it is necessary to have an understanding of:

- The Network Manager event source data that is available for use in the probe rules file
- The set of alerts.status fields that can be populated with event data from Network Manager
- The data mapping between the Network Manager and alerts.status fields

For information about the syntax used in probe rules files, see the *IBM Tivoli Netcool/OMNIbus Probe and Gateway Guide* in the Tivoli Netcool/OMNIbus information centre at http://publib.boulder.ibm.com/infocenter/tivihelp/v8r1/topic/com.ibm.tivoli.nam.doc/welcome_ob.htm.

Example of rules file processing

This example shows how source data from Network Manager is processed by the rules file to generate the output data that is inserted in the alerts.status table.

The following sample code shows a Network Manager event data record that is passed to the Probe for Tivoli Netcool/OMNIbus for processing. In this record, a resolution event was created when **ncp_ctrl** started the **ncp_store** process.

```
{
 EventName='ItnmServiceState';
 Severity=1;
 EntityName='BACKUP';
 Description='ncp_store process [15299] has started';
 ExtraInfo={
  EVENTTYPE=2;
  SOURCE='ncp_ctrl';
  ALERTGROUP='ITNM Status';
  EVENTMAP='ItnmStatus';
  SERVICE='ncp_store';
  PID=15299;
  };
}
```

The following excerpt from the probe rules file shows the syntax used to process and map these input fields to alerts.status fields:

```
...
    #
    # populate some standard fields
    #
    @Severity = $Severity
    @Summary = $Description
    @EventId = $EventName
    @Type = $ExtraInfo_EVENTTYPE
    @AlertGroup = $ExtraInfo_ALERTGROUP
    @NmosEventMap = $ExtraInfo_EVENTMAP
    @Agent = $ExtraInfo_SOURCE

    if (exists($ExtraInfo_ACCESSIPADDRESS))
    {
        @Node = $ExtraInfo_ACCESSIPADDRESS
    }
    else
    {
        @Node = $EntityName
    }

    #
    # Stamp the event with the name of its originating domain
    #
    @NmosDomainName = $Domain
    @Manager = "ITNM"
    @Class = 8000

    #
    # populate fields for RCA
    #
    @LocalNodeAlias = @Node

...

    #
    # Now set the AlertKey and Identifier
    #
    if (match(@AlertGroup, "ITNM Status"))
    {
```

```
        switch ($EventName)
        {
            case ...
...
            case "ItnmServiceState":
                @LocalPriObj = $ExtraInfo_SERVICE
...
            case ...
....
        }
    }

    #
    # Both the Identifier and the AlertKey contain the domain name. This ensures
    # that in a multi-domain setup, events are handled on a per-domain basis
    #

    #
    # Include the LocalPriObj in the AlertKey or the link-downs on
    # all interfaces will cleared by a link-up on any interface
    #
    @AlertKey = $EntityName + @LocalPriObj + "->" + $EventName + @NmosDomainName

    #
    # Set up deduplication identifier and include the LocalPriObj
    # so we can correctly handle de-duplication of events raised on interfaces
    #
    @Identifier = $EntityName + @LocalPriObj + "->" + $EventName + @Type + @NmosDomainName
}
```

When rules file processing is complete, the output data that is forwarded to the ObjectServer takes the following form:

```
CMonitorProbeApp::ProcessStatusEvent
{
 AlertGroup='ITNM Status';
 EventId='ItnmServiceState';
 Type=2;
 Severity=1;
 Summary='ncp_store process [15299] has started';
 Node='BACKUP';
 NmosDomainName='PRIMARY';
 LocalNodeAlias='BACKUP';
 LocalPriObj='ncp_store';
 LocalRootObj='';
 RemoteNodeAlias='';
 AlertKey='BACKUPncp_store->ItnmServiceStateVIRTUAL';
 Identifier='BACKUPncp_store->ItnmServiceState2VIRTUAL';
 Class=8000;
 Agent='ncp_ctrl';
 LastOccurrence=1267122089;
}
```

Based on the rules file processing in this example, it can be seen that the Network Manager input fields map to the alerts.status fields as follows:

Network Manager field	**alerts.status field**
EventName	EventId
Severity	Severity
EntityName	Node
Description	Summary
ExtraInfo->EVENTTYPE	Type
ExtraInfo->SOURCE	Agent
ExtraInfo->ALERTGROUP	AlertGroup
ExtraInfo->EVENTMAP	NmosEventMap

Network Manager field	alerts.status field
ExtraInfo->SERVICE	LocalPriObj

Network Manager event data fields

When events are generated in Network Manager, the event data is inserted into a number of fields (or columns) in the Network Manager tables. Although each event uses only a subset of the possible fields, a number of fields are common to all event types.

The following table lists all the Network Manager field names that are available for use in the probe rules file, and describes the event data stored in each field. The table also identifies which of the Network Manager fields are common to all events, and therefore always available in the rules file.

Table 62. Network Manager fields that populate events

Network Manager field name	Field content	Always available?
Description	A brief description of the event.	Yes
Domain	The current domain. If Network Manager is configured for failover mode, this will be the primary domain.	Yes (provided the map file is not modified)
EntityName	For network events, this is the entityName field from the NCIM entityData table for the entity against which the event is raised. For status events, this is always the name of the domain about which the event is generated.	Yes
EventName	The event identifier. For example, ItnmDiscoPhase.	Yes
ExtraInfo_ACCESSIPADDRESS	If the main node or interface entity identified by the EntityName input field has a directly-accessible IP address (the accessIPAddress field from the NCIM interface or chassis tables), then it is supplied here. Applicable to network events only.	No
ExtraInfo_AGENT	The agent responsible for a discovery agent (ItnmDiscoAgentStatus) event.	Yes (for ItnmDiscoAgentStatus events)
ExtraInfo_ALERTGROUP	The alert group of the event. For Network Manager status events, the alert group is `ITNM Status`, and for network events, the value is `ITNM Monitor`.	Yes
ExtraInfo_ENTITYCLASS	The name of class assigned to the entity, as identified the NCIM entityClass and classMembers tables.	Yes (for network and ItnmEntityCreation events)
ExtraInfo_ENTITYTYPE	The type of the entity, as defined in the NCIM entityType table.	Yes (for network events)

Table 62. Network Manager fields that populate events (continued)

Network Manager field name	Field content	Always available?
ExtraInfo_LocalPriObj	Provides a value for the LocalPriObj field in the alerts.status record. This field has the same value as the deprecated ExtraInfo_EventSnmpIndex field, except that it is prefixed by an identifier for the MIB entity being polled; for example ifEntry, bgpPeerEntry.	Yes (for network events)
ExtraInfo_EVENTTYPE	The type of the event raised by Network Manager. The values are as follows: • 1: Problem • 2: Resolution • 13: Information	Yes
ExtraInfo_FINDER	The finder responsible for a discovery finder (ItnmDiscoFinderStatus) event.	Yes (for ItnmDiscoFinderStatus events)
ExtraInfo_ifIndex	For events raised against an interface with an ifIndex value in the NCIM interface table, that value is given here. Applicable only to network events against interfaces.	No
ExtraInfo_IFALIAS	For events raised against interfaces, this field contains the ifAlias value, if known. Applicable only to network interface polls.	No
ExtraInfo_IFDESCR	For events raised against interfaces, this field contains the ifDescr value, if known. Applicable only to network interface polls.	No
ExtraInfo_IFNAME	For events raised against interfaces, this field contains the ifName value, if known. Applicable only to network interface polls.	No
ExtraInfo_IFTYPESTRING	For events raised against interfaces, this field contains the string representation of the ifType value. Applicable only to network interface polls.	No
ExtraInfo_MAINNODEADDRESS	The management interface of the main node containing the entity, as identified by the accessIPAddress field of the NCIM chassis table. Applicable only to network and ItnmEntityCreation events.	Yes (for network events)
ExtraInfo_MAINNODEENTITYID	The entityId field from the NCIM entityData table for the main node, as identified by the accessIPAddress field of the NCIM chassis table. Applicable only to network events.	Yes (for network events)
ExtraInfo_MAINNODEENTITYNAME	The entityName field from the NCIM entityData table for the main node, as identified in NCIM. Applicable only to network events.	Yes (for network events)
ExtraInfo_MONITOREDENTITYID	The entityId field from the NCIM entityData table for the entity against which the event is raised. Applicable only to network and ItnmEntityCreation events.	No
ExtraInfo_MONITOREDINSTID	A record in the ncpolldata.monitoredInstance table.	No

Table 62. Network Manager fields that populate events (continued)

Network Manager field name	Field content	Always available?
ExtraInfo_NEWPHASE	The discovery phase that has started. Applicable only to discovery phase (ItnmDiscoPhase) events.	Yes (for discovery phase events)
ExtraInfo_OLDPHASE	The discovery phase that has completed. Applicable only to discovery phase (ItnmDiscoPhase) events.	Yes (for discovery phase events)
ExtraInfo_POLICYNAME	The name of the polling policy that resulted in the event.	Yes (for network events)
ExtraInfo_PID	The process ID of the affected Network Manager service. Applicable only to ItnmServiceState events.	Yes (for service state events)
ExtraInfo_REMOTEDOMAIN	The name of the remote domain. Applicable only to ItnmFailoverConnection events.	Yes (for failover connection events)
ExtraInfo_sysContact	If available, the sysContact value is given for ItnmEntityCreation events only.	No
ExtraInfo_sysLocation	If available, the sysLocation value is given for ItnmEntityCreation events only	No
ExtraInfo_sysObjectId	If available, the sysObjectId value is given for ItnmEntityCreation events only	No
ExtraInfo_SERVICE	The name of the affected Network Manager service. Applicable only to ItnmServiceState events.	Yes (for service state events)
ExtraInfo_SNMPSTATUS	A numerical SNMP status code.	Yes (for NmosSnmpPollFail events)
ExtraInfo_SNMPSTATUSSTRING	A human-readable indication of the SNMP failure state.	Yes (for NmosSnmpPollFail events)
ExtraInfo_SOURCE	The name of the process from which the event originated.	Yes
ExtraInfo_STITCHER	The stitcher responsible for a discovery stitcher (ItnmDiscoStitcherStatus) event.	Yes (for ItnmDiscoStitcherStatus events)
Severity	The severity level of the event. The severity is a non-zero value.	Yes

alerts.status fields used by Network Manager

The alerts.status table in the ObjectServer contains status information about problems that have been detected by probes.

A subset of the standard alerts.status fields is populated with Network Manager event data. Additionally, a set of dedicated alerts.status fields are reserved to hold data that is specific to Network Manager. The dedicated alerts.status field names are identifiable by the prefix `Nmos`.

The following table describes the alerts.status fields that are populated by Network Manager fields. Some of these alerts.status fields are allocated default values from within the probe rules file. (Avoid modifying these default values.)

Table 63. alerts.status fields used by Network Manager

alerts.status field	Data type	Description	Network Manager field name/Default value in rules file
Agent	varchar(64)	The name of the process that generated the event. You can use this field to filter an AEL to display only events of a specific type; for example, only discovery events (with a value of ncp_disco).	ExtraInfo_SOURCE
AlertGroup	varchar(255)	Used to group events by type. Values supplied by default from Network Manager events are either `ITNM Monitor` for network events, or `ITNM Status` for status events.	ExtraInfo_ALERTGROUP
AlertKey	varchar(255)	A text string concatenating several elements relating to the event. Elements can include the event ID, domain, phase, and process name. Allows problem and resolution events to be matched.	This value is generated from the input to ensure appropriate matching of problem and resolution events within the ObjectServer.
Class	integer	The alert class asigned to the Probe for Tivoli Netcool/OMNIbus.	A value of `8000` is reserved for events generated by Network Manager.
EventId	varchar(255)	The type of event (for example, SNMPTRAP-linkDown). The Event Gateway uses this value to look up the event map, and to determine the precedence of events.	EventName
ExpireTime	integer	The expiry time of the event in the database. Not currently used by Network Manager.	
FirstOccurrence	time	A timestamp indicating when the event first occurred.	
Identifier	varchar(255)	A unique value for each type of event on a given entity (for example, a LinkDown event on a specific device interface). This identifier controls deduplication.	This value is generated from the input to ensure appropriate deduplication of events in the ObjectServer. In the rules file, the identifier is constructed as a concatenation of field values.
LastOccurrence	time	A timestamp indicating when the event last occurred.	
LocalNodeAlias	varchar(64)	The IP or DNS address of the device. This value usually refers to the chassis, but for pingFails only, can correspond to the interface.	For network events, this field is set to the same value as the Node field. No value is set for status events, to ensure that they are not fed back into Network Manager through the Event Gateway.

Table 63. alerts.status fields used by Network Manager (continued)

alerts.status field	Data type	Description	Network Manager field name/Default value in rules file
LocalPriObj	varchar(255)	The specific entity for which the event is generated; for example, the ifIndex, ifDescr, or ifPhysAddress field value.	ExtraInfo_AGENT or ExtraInfo_FINDER or ExtraInfo_ifIndex or ExtraInfo_NEWPHASE or ExtraInfo_SERVICE or ExtraInfo_STITCHER The ExtraInfo_ifIndex value is shown using the syntax `ifEntry.<ifIndex>`; for example, `ifEntry.12`.
LocalRootObj	varchar(255)	The container of the entity referenced in the LocalPriObj field. This need not be the chassis, but could, for example, be slot in a chassis. The chassis can still be identified using LocalNodeAlias.	
LocalSecObj	varchar(255)	The secondary object referenced by the event.	ExtraInfo_OLDPHASE
Manager	varchar(64)	A descriptive name that identifies the system that forwarded the events.	A value of `ITNM` is used for events generated by Network Manager V3.8, or later. A value of `Omnibus` is used in earlier versions.
NmosCauseType	integer	The event state. Populated by the NMOS gateway. The possible values are as follows: • 0: Unknown • 1: Root Cause • 2: Symptom	
NmosDomainName	varchar(64)	The name of the Network Manager network domain that raised the event. The name of the primary domain is used in failover mode. By default, this field is populated only for events that are generated by Network Manager. To populate this field for other event sources, such as those from other probes, you must modify the rules files for those probes. This field is populated by the Event Gateway if an event is matched to an entity in a domain.	Domain

Table 63. alerts.status fields used by Network Manager (continued)

alerts.status field	Data type	Description	Network Manager field name/Default value in rules file
NmosEntityId	integer	The unique Object ID that identifies the topology entity with which the event is associated. This field is similar to the NmosObjInst field but contains more detailed information. For example, this field can include the ID of an interface within a device. For events generated by the Polling engine, the NmosEntityId field is populated in the probe rules file. For all other events, this field is populated by the gateway when the entity is identified.	ExtraInfo_MONITOREDENTITYID
NmosEventMap	varchar(64)	The event map name and optional precedence for the event, which indicates how Network Manager should process the event; for example, PrecisionMonitorEvent.910. The optional precedence number can be concatenated to the end of the value, following a period (.). If the precedence is not supplied, it is set to 0. **Note:** This value can be overridden by an explicit insertion into the Event Gateway config.precedence table, which provides the same data.	
NmosManagedStatus	integer	The managed status of the network entity for which the event was raised. When a network entity is unmanaged, the Network Manager polls are suspended and events from other sources are tagged as unmanaged. This field allows you to filter out events from unmanaged entities. The possible values for this field are as follows: • 0: Managed • 1: Operator unmanaged • 2: System unmanaged • 3: Out of scope	
NmosObjInst	integer	The unique Object ID that identifies the containing topology chassis entity with which the event is associated. Populated by the NMOS gateway. **Tip:** This field can be used to detect whether the event has been passed for event enrichment.	
NmosSerial	integer	The serial number of the event that is suppressing the current event. Populated by the NMOS gateway.	

Table 63. alerts.status fields used by Network Manager (continued)

alerts.status field	Data type	Description	Network Manager field name/Default value in rules file
Node	varchar(64)	The device from which the event originated. If an event is raised against an entity with an accessible IP address, the IP address is used. Otherwise, the entityName value from NCIM is used. By default, Node has the same value as LocalNodeAlias.	ExtraInfo_ACCESSIPADDRESS or EntityName The EntityName value maps to the Node field only if the ExtraInfo_ACCESSIPADDRESS input field is empty.
NodeAlias	varchar(64)	The IP address of the main node, if available.	ExtraInfo_MAINNODEADDRESS
RemoteNodeAlias	varchar(64)	The network address of a remote node, where relevant. For example: • A blank value (where an interface has gone down) • A neighbouring address (where a connected interface has gone down) • The polling station (for a ping failure event)	
Serial	incr	A unique ID per event per ObjectServer instance. Where primary and backup ObjectServers are configured, the ObjectServers will have different serial numbers for the same event.	
ServerName	varchar(64)	The name of the originating ObjectServer.	
ServerSerial	integer	The Serial number of the event in the originating ObjectServer. Where primary and backup ObjectServers are configured, the ObjectServers will have different serial numbers for the same event. If the event originated in the current ObjectServer, the ServerSerial value is the same as the Serial value.	
Severity	integer	The severity level of the event stored in the ObjectServer. The default values are as follows: • 0: Clear (GREEN) • 1: Indeterminate (PURPLE) • 2: Warning (BLUE) • 3: Minor (YELLOW) • 4: Major (ORANGE) • 5: Critical (RED)	Severity
StateChange	time	A timestamp indicating when the event was last modified. This field can be used to determine whether a process is modifying an event after it has been added to the ObjectServer.	
Summary	varchar(255)	A textual description of the event.	Description

Table 63. alerts.status fields used by Network Manager (continued)

alerts.status field	Data type	Description	Network Manager field name/Default value in rules file
Tally	integer	A count of the number of times that an event has occurred. This value is displayed in the Count column in the event list or AEL, and in the Occurred column in the mojo.events table.	
Type	integer	The type of the alert. The values of particular relevance to Network Manager are • 1: Problem • 2: Resolution • 13: Information	ExtraInfo_EVENTTYPE

For more information about the alerts.status table, see the *IBM Tivoli Netcool/OMNIbus Administration Guide* in the Tivoli Netcool/OMNIbus information centre at http://publib.boulder.ibm.com/infocenter/tivihelp/v8r1/topic/com.ibm.tivoli.nam.doc/welcome_ob.htm.

Appendix F. Network Manager event categories

The events that are raised by Network Manager fall into two categories: events about the network being monitored and events about Network Manager processes.

These events are stored in the Tivoli Netcool/OMNIbus ObjectServer. The Probe for Tivoli Netcool/OMNIbus (**nco_p_ncpmonitor**) is used to process and forward the event data to the alerts.status table in the ObjectServer.

The following figure shows the flow of events from Network Manager to the ObjectServer.

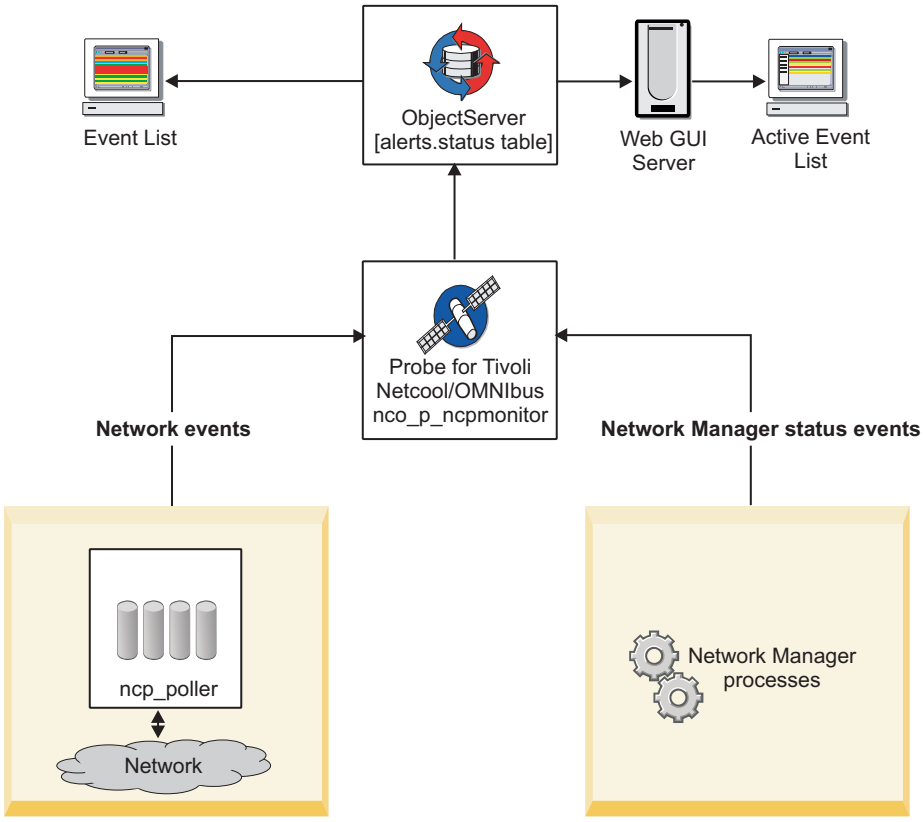


Figure 15. Flow of events from Network Manager to Tivoli Netcool/OMNIbus

Related reference:

"Default event maps" on page 98
Network Manager provides a default set of event maps. Use this information to understand which default event maps are available and what each event map does, and to understand how legacy event maps delegate to 3.9 event maps.

© Copyright IBM Corp. 2006, 2016

Network Manager network events

The Polling engine, **ncp_poller**, generates events about the state of the network. These events can be used to identify network problems, and are configurable by using the Network Polling GUI (go to **Administration** > **Network** > **Network Polling**). These events are known as network events and have the alerts.status AlertGroup field value of `ITNM Monitor`.

Each network event is raised on a single entity, such as an interface or a chassis, and the event data is dependent on the type of poll. When network events are forwarded to the ObjectServer for insertion into the alerts.status table, they are allocated an AlertGroup value of `ITNM Monitor`.

An unlimited set of event identifiers is available for network events. Events that are generated when an SNMP poll fails are specifically allocated an EventID value of `NmosSnmpPollFail` in the alerts.status table.

Network events in the ObjectServer are pulled back into Network Manager through the Event Gateway to perform event enrichment, including root cause analysis.

Network Manager status events

Network Manager can generate events that show the status of various Network Manager processes. These events are known as Network Manager status events and have the alerts.status AlertGroup field value of `ITNM Status`.

When these status events are forwarded to the ObjectServer for insertion into the alerts.status table, they are allocated an AlertGroup value of `ITNM Status`.

Status event types

A set of event identifiers is used to identify Network Manager status events by type. The following list identifies the EventId values that are inserted in the alerts.status table, and describes how each associated status event is generated.

ItnmDatabaseConnection

This type of event is generated to indicate loss of connection to NCIM. This event is generated by the managed status polling thread in the **ncp_model** process. The raising of this event depends on the time period configured in the managed status polling interval in model. A problem event is raised if the connection is lost, and a corresponding resolution event is raised if the connection is restored, or at startup to clear any failures from a previous operation. This event type allows the backup domain to take over when failover is configured. The virtual domain process reacts to this event as defined in the filter for NCIM in the `NCHOME/etc/precision/VirtualDomainSchema.cfg` file.

ItnmDiscoAgentStatus

This type of event is generated by **ncp_disco** when a discovery agent transitions to a new state. At the end of a discovery, an information event is forwarded to the ObjectServer, for each agent that was used during the discovery.

You can use this information to identify the state of each agent. In the alerts.status table, the LocalPriObj field is used to store the name of the agent.

Discovery agent events in the ObjectServer are overwritten when a subsequent discovery is run.

ItnmDiscoFinderStatus

This type of event is generated by **ncp_disco** when a discovery finder transitions to a new state. At the end of a discovery, an information event is forwarded to the ObjectServer, for each finder that was used during the discovery.

You can use this information to identify which finders are running and their state. In the alerts.status table, the LocalPriObj field is used to store the name of the finder.

Discovery finder events in the ObjectServer are overwritten when a subsequent discovery is run.

ItnmDiscoPhase

This type of event is generated by **ncp_disco** when the discovery process transitions to a new phase. At the end of the discovery, five information events should be present in the ObjectServer, to show the looped transitions from phase 0 (standby) to phases 1, 2, and 3 (data collection) to phase -1 (data processing). An event is raised for each of the following phase changes in a single discovery:

- 0 to 1
- 1 to 2
- 2 to 3
- 3 to -1
- -1 to 0

You can use this information to determine the length of each phase. In the alerts.status table, the LocalPriObj field is used to store the phase to which the discovery is transitioning, and the LocalSecObj field stores the previous phase of the discovery.

Tip: The string values for the phases are also shown in the discovery log file when the **ncp_disco** process is run in debug mode.

Discovery phase events in the ObjectServer are overwritten when a subsequent discovery is run.

ItnmDiscoStitcherStatus

The discovery process is made up of a data collection stage and a data processing stage, during which the topology is created. ItnmDiscoStitcherStatus events are generated by the Discovery engine, **ncp_disco**, when a major phase is reached in the data processing stage. At the end of the discovery, an information event is forwarded to the ObjectServer, for each major discovery stitcher that was used during the discovery.

You can use this information to identify what phase in the data processing stage the discovery is in. In the alerts.status table, the LocalPriObj field is used to store the name of the stitcher corresponding to this phase.

ItnmDiscoStitcherStatus events are raised when the following stitchers begin executing:

- BuildFinalEntityTable
- BuildContainment
- BuildLayers

- MergeLayers
- PostLayerProcessing

Subsequently events are raised during the topology creation phase when the following stitchers are run.

- CreateScratchTopology
- PostScratchProcessing
- SendTopologyToModel

Discovery stitcher events in the ObjectServer are overwritten when a subsequent discovery is run.

ItnmEntityCreation

If configured in the `$NCHOME/etc/precision/ModelSchema.cfg` file, this type of information event is generated by **ncp_model**, for each new chassis or IP interface entity (EntityType = 1) that is inserted into the NCIM database.

You can configure `ModelSchema.cfg` by setting the value of the RaiseEntityEvent column to 1 in the INSERT statement for the model.config table. For example:

```
create table model.config
(
LingerTime int not null primary key,           // default value 3 (discoveries)
RaiseEntityEvent int type boolean not null,    // default value 0 ( off )
DiscoveryUpdateMode int not null,              // default value 0 - full discovery,
                                               //               1 - partial
unique(LingerTime)
);
insert into model.config values (3, 1, 0);
```

Note: For the configuration changes to take effect and enable the events, the **ncp_model** process must be restarted. The process reads the configuration settings at start-up.

ItnmEntityDeletion

If configured in the `$NCHOME/etc/precision/ModelSchema.cfg` file, this type of information event is generated by **ncp_model**, for each chassis or IP interface entity (EntityType = 1) that is deleted from the NCIM database.

You can configure `ModelSchema.cfg` by setting the value of the RaiseEntityEvent column to 1 in the INSERT statement for the model.config table, as shown in the preceding description for the ItnmEntityCreation EventId.

ItnmFailover

This type of event is generated by **ncp_virtualdomain** when a Network Manager domain within a failover pair fails over or fails back.

A problem event is generated when failover occurs and a resolution event is generated on failback.

In the alerts.status table, the Summary field description indicates whether the domain is the primary or backup, and whether it is in an active or a standby mode.

ItnmFailoverConnection

This type of event is generated by **ncp_virtualdomain** to indicate when the backup domain in a failover pair connects to, or disconnects from, the primary domain.

When Network Manager runs in failover mode, a resolution event is generated when the primary and backup domains set up their TCP socket connection. This socket connection is required to transfer the topology

updates from the primary domain because the discovery process (**ncp_disco**) does not run in the backup domain. If the connection is subsequently lost, a problem event is generated.

Note: The status of the connection does not determine whether failover is triggered. Failover is triggered only when health check events are transferred (via the ObjectServer) across domains, and provided a socket connection has, at some point, been established.

ItnmHealthChk

Health check events govern Network Manager failover. Each domain in the failover pair generates health check resolution events while that domain is healthy.

Health check problem events for a domain can be generated in two ways:

- By the local domain: The local domain detects a failure of one of its processes, as configured in the `$NCHOME/etc/precision/VirtualDomainSchema.cfg` file.
- By the remote domain: One domain detects that the other domain has not generated a health check resolution event in the configured amount of time, and generates a synthetic health check problem event on behalf of the remote domain.

When a health check problem event is generated for the primary domain, failover is initiated, and the backup domain becomes active.

Health check events were previously allocated an EventID value of NcpHealthChk. For compatibility with earlier versions of Network Manager, you can substitute NcpHealthChk in place of ItnmHealthChk in the probe rules file.

Note: Health check events are handled by the Network Manager Event Gateway, which requires the Node value to be the domain to which the event refers. This need not be the domain raising the event, since one domain can raise failure events on behalf of the other.

ItnmMaintenanceState

If configured in the `$NCHOME/etc/precision/ModelSchema.cfg` file, this type of event is generated by the Topology manager, **ncp_model**, for maintenance status changes to a chassis or an IP interface.

You can configure `ModelSchema.cfg` by setting the value of the RaiseEntityEvent column to 1 in the INSERT statement for the model.config table, as shown in the preceding description for the ItnmEntityCreation event.

A problem event is generated when the chassis or IP interface entity is in maintenance, and a resolution event is generated when the entity is out of maintenance.

Note: An individual interface event is sent only if the change does not apply at the chassis level; when a device changes, a chassis event and a series of interface events are not collectively generated.

ItnmServiceState

This type of event is generated when a process starts or ends, and signifies whether a process has failed to start or has stopped during runtime. (Note that process state events are not generated when processes are stopped at system shutdown.)

A resolution event is generated when **ncp_ctrl** starts a process. If a process fails to start or if it stops during runtime, a problem event is generated.

In the alerts.status table, the Summary field description includes the process name, the PID, and an indication of whether the process has:

- Started (and successfully initialized)
- Stopped (that is, it has been deleted from the **ncp_ctrl** database table named services.inTray)
- Terminated (that is, it stopped, but will be restarted by **ncp_ctrl**)
- Failed to start
- Failed and will not be restarted (that is, it stopped and the number of retries configured for the process has been exceeded)

ItnmTopologyUpdated

This type of information event is generated by **ncp_model** when the update of the NCIM topology database is completed at the end of a discovery cycle. This information is useful if you intend to program scripts or procedures to run after the NCIM database is updated.

Note: If the feedback option is on, or large subnets are pinged, there might be multiple discovery cycles and thus multiple events of this type, one event for each discovery cycle. To determine if discovery has finally finished, the following OQL query can be made to the Ping Finder service:

```
select * from pingFinder.status where m_Completed <> 1;
```

This query looks for any subnets that the Ping finder is still pinging. If there are no outstanding ping sweeps and the discovery is in phase 0, this means that the discovery is complete.

Appendix G. Polling databases

Use this information to understand the structure of databases used for polling.

NCMONITOR databases

The NCMONITOR schema hosts a number of databases used by polling.

SNMP tables for polling in the ncmonitor database

The SNMP tables in the ncmonitor database are used by the polling engine, ncp_poller, to store information on how to access each discovered device using SNMP.

Both the ncp_dh_snmp and ncp_poller processes use the ncmonitor database. However, only the ncp_dh_snmp process populates the database; the ncp_poller process treats it as read-only. Hence, you must have discovered a device using SNMP to monitor it using SNMP.

The ncmonitor database is defined in `$NCHOME/etc/precision/DbLogins.DOMAIN.cfg`, where *DOMAIN* is the domain that contains the discovered devices.

The ncmonitor database has the following tables:

- ncmonitor.snmpTarget
- ncmonitor.snmpAccess
- ncmonitor.snmpv1Sec
- ncmonitor.snmpv3Sec
- ncmonitor.snmpUser

ncmonitor.snmpTarget table

The snmpTarget table lists each IP address that Network Manager recognizes.

Table 64. ncmonitor.snmpTarget database table

Column name	Constraints	Data type	Description
targetid	• PRIMARY KEY • NOT NULL	Text	Unique identifier for the target.
netaddr		Text	IP address of the target.
readaccessid	FOREIGN KEY	Text	Refers to the snmpaccess table. Provides access details used to perform SNMP Get and GetNext operations for this target.
writeaccessid	FOREIGN KEY	Text	Refers to the snmpaccess table. Provides access details used to perform SNMP Set operations for this target.

© Copyright IBM Corp. 2006, 2016

Table 64. ncmonitor.snmpTarget database table (continued)

Column name	Constraints	Data type	Description
snmpgetbulk		Boolean integer	Flag indicating whether GetNext operations will be attempted when appropriate; for example, when using SNMPv2 or SNMPv3, and performing a table walk.
snmpthrottleid		Text	Throttling details used to control the rate at which requests will be made to this target when performing table walk operations.
createtime		Text	Timestamp recording the time this target was created.
lastupdate		Text	Timestamp recording the time any detail for this target was last modified.
domain		Text	Domain to which this target belongs.

ncmonitor.snmpAccess table

The snmpAccess table provides details of SNMP access.

Table 65. ncmonitor.snmpAccess database table

Column name	Constraints	Data type	Description
accessid	• PRIMARY KEY • NOT NULL	Text	Unique identifier for these SNMP access details.
version		Enumerated value	SNMP version to be used. Possible value are: • 0: SNMPv1 • 1: SNMPv2 • 3: SNMPv3
remoteport		Integer	UDP port to which SNMP packets will be sent.
retries		Integer	Number of retries before giving up.
timeout		Integer	Number of milliseconds before retrying an SNMP request .
accesslevel		Enumerated value	Flag indicating level of access provided. Possible values are: • 1: read • 2: write

ncmonitor.snmpv1Sec table

The snmpv1Sec table is populated only for rows in the snmpAccess table that relate to SNMPv1 and SNMPv2.

Table 66. ncmonitor.snmpv1Sec database table

Column name	Constraints	Data type	Description
accessid	FOREIGN KEY	Text	Refers to the details for which SNMPv1 or SNMPv2-specific detail is being provided.
community		Text	Community string to use when sending requests using these details.
encrypted		Boolean integer	Flag indicating whether the community string is encrypted. Possible values are: • `0`: not encrypted • 1: encrypted

ncmonitor.snmpv3Sec table

The snmpv3Sec table is populated only for rows in the snmpAccess table that relate to SNMPv3.

Table 67. ncmonitor.snmpv3Sec database table

Column name	Constraints	Data type	Description
accessid	FOREIGN KEY	Text	Refers to the details for which SNMPv3-specific detail is being provided.
userid	FOREIGN KEY	Text	Refers to the userid field in the snmpusmuser table. This is the user to use when sending SNMP requests using these details.
securitylevel		Enumerated value	Flag indicating SNMPv3 security level. Possible values are: • `noAuthNoPriv` • `authNoPriv` • `authPriv`
defaultcontext		Text	SNMPv3 contextName to be used when not explicitly specified by a discovery agent.

ncmonitor.snmpUser table

The snmpUser table provides a list of SNMP user details to be used by the SNMPv3 protocol.

Table 68. ncmonitor.snmpUser database table

Column name	Constraints	Data type	Description
userid	• PRIMARY KEY • NOT NULL	Text	Unique identifier for this user.
username		Text	USM username.
authpass		Text	Authentication password.

Table 68. ncmonitor.snmpUser database table (continued)

Column name	Constraints	Data type	Description
privpass		Text	Privacy password. This is used for encrypting the SNMPv3 payload.
authtype		Text	Encryption method to be used for the SNMPv3 authentication header.
privtype		Text	Encryption method to be used for the payload.
encrypted		Boolean integer	Flag indicating authpass and authtype fields are encrypted. Possible values are: • 0: not encrypted • 1: encrypted

Ping polling status tables

The NCMONITOR ping polling status tables enable diagnostic operations to be performed on network ping polling.

expectedIps table

The expectedIps table contains a list of IP addresses expected to be discovered by Network Manager for a particular domain. It is populated using the ncp_upload_expected_ips.pl script.

The following table lists the columns in the expectedIps table.

Table 69. ncmonitor.expectedIps database table

Column name	Description
ip	A dot-notation IPv4 address that is expected to have been discovered and added to the NCIM topology database.
domainMgrId	The ID of the relevant domain, as found in the NCIM topology database domainMgr table.

pollLog table

The pollLog table stores the latest snapshot of the status of the Polling engine. It is populated using the ncp_ping_poller_snapshot.pl script which queries ncp_poller, and transfers the results to this table.

Each row in this table corresponds to a single entity that is within the defined scope of a single active polling policy. The fields in the table can be divided into three conceptual groupings:

"Entity information" on page 217
This entity information can be used to cross-reference with other NCIM topology database tables.

"Managed status information" on page 217
This is the managed status being applied by the poll policy.

"Latest poll state information" on page 218
This is the latest poll state for the current entity and policy.

Entity information

Fields in the pollLog table that store entity information are described below.

Table 70. ncmonitor.pollLog database table entity information fields

Column name	Description
entityId	ID of the entity to ping, as defined in the NCIM topology database entityData table.
policyId	ID of the relevant ping policy, as defined in the NCMONITOR policy table.
mainNodeEntityId	ID of the main node that the entity belongs to, as defined in the NCIM topology database entityData table. For Chassis Ping polls, this is the same as the entityId. For Interface Ping polls, this is the ID of the main node containing the interface.
entityType	As defined in the NCIM topology database entityType table, this field can take one of the following values: • 1: Chassis Ping polls • 2 Interface Ping polls
ip	IP address to which the ICMP ping packet was sent. This is the accessIPAddress of the interface or chassis entity identified by entityId.
mainNodeAddress	IP address of the main node that the entity belongs to, as found in the NCIM topology database entityData table. This is the accessIPAddress of the chassis entity identified by mainNodeEntityId.
ifIndex	The ifIndex of the relevant interface might be available for Interface Ping polls. This can be NULL for any ping poll.
domainMgrId	ID of the relevant domain, as found in the NCIM topology database domainMgr table.

Managed status information

Fields in the pollLog table that store managed status information are described below.

Table 71. ncmonitor.pollLog database table managed status information fields

Column name	Description
isManaged	Indicates whether the entity is being polled by this policy? • 0: False • 1: True
entityStatus	Managed status of the entity identified by entityId that is being used by the poller. The value of this field is set to 0 if managed, whether explicitly listed in the NCIM managedStatus table or not. **Note:** This is the status at the time of the snapshot, and therefore can differ from the contents of the NCIM managedStatus table if it is dynamically altered.

Table 71. ncmonitor.pollLog database table managed status information fields (continued)

Column name	Description
mainNodeStatus	Managed status of the entity identified by mainNodeEntityId that is being used by the poller. Interfaces within an unmanaged main node are also unmanaged. The value of this field is set to 0 if managed, whether explicitly listed in the NCIM managedStatus table or not. **Note:** This is the status at the time of the snapshot, and therefore can differ from the contents of the NCIM managedStatus table if it is dynamically altered.
entityChangeTime	Timestamp of the last change to the entityStatus field. Defaults to a zero timestamp if unused.
mainNodeChangeTime	Timestamp of the last change to the mainNodeStatus field. Defaults to a zero timestamp if unused.

Latest poll state information

Fields in the pollLog table that store latest poll state information are described below.

Table 72. ncmonitor.pollLog database table latest poll state information fields

Column name	Description
lastPollFailure	Last time at which a ping poll failure event was raised. Defaults to a zero timestamp if no poll failures have been raised since the poller was started.
lastPollInterval	Duration of the last complete polling cycle, in seconds. This field is NULL if the policy is not actively monitoring the entity or a complete poll cycle has not finished since the poller started. **Note:** The system must be on the third polling interval when the snapshot is taken.
timeSinceLastPoll	Number of seconds since the last poll, at the time the snapshot was taken.
snapshotTime	Time at which the data was retrieved from the poller. This is a zero timestamp if the policy is not actively monitoring the entity.

pollLogSummary table

This table stores a summary of the results for each snapshot written to the pollLog table for a domain, generated using the views listed in the following sections. It is populated using the ncp_ping_poller_snapshot.pl script which queries the poller, and transfers the results to this table.

The following table lists the columns in the pollLogSummary table.

Note: The ncp_ping_poller_snapshot.pl script never clears out existing data from this table, so the table can grow. If required, data that is no longer of interest can be removed by filtering against the domainMgrId or the summaryTimestamp fields.

Table 73. ncmonitor.pollLogSummary database table

Column name	Description
domainMgrId	ID of the relevant domain, as found in the NCIM domainMgr table.
domainName	Name of the domain identified by the domainMgrId.
undiscoveredIps	Count of IP addresses returned from the undiscoveredIps view for this domain after the snapshot was loaded to the pollLog.
unmonitoredIps	Count of IP addresses returned from the unmonitoredIps view for this domain after the snapshot was loaded to the pollLog.
unmanagedIps	Count of IP addresses returned from the unmanagedIps view for this domain after the snapshot was loaded to the pollLog.
unpolledFor15MinutesIps	Count of IP addresses returned from the unpolledFor15MinutesIps view for this domain after the snapshot was loaded to the pollLog.
delayedPollPolicies	Count of IP addresses returned from the delayedPollPolicies view for this domain after the snapshot was loaded to the pollLog table.
summaryTimestamp	Timestamp indicating when the summary was generated.

undiscoveredIps view

The undiscoveredIps view lists any IP addresses that were are not discovered by Network Manager and therefore are not listed in the NCIM topology database, but that you expected to discover. The IP addresses listed in this table are those that were loaded into the expectedIps table but are not present in NCIM.

The following table lists the columns in the undiscoveredIps view.

Table 74. ncmonitor undiscoveredIps view

Column name	Description
ip	A dot-notation IPv4 address that is expected to have been discovered and added to the NCIM topology database.
domainMgrId	The ID of the relevant domain, as found in the NCIM topology database domainMgr table.
domainName	The name of the domain identified by the domainMgrId column..

unmonitoredIps view

The unmonitoredIps view uses the latest poller snapshot from the pollLog table to list any IP addresses from the expectedIps table that are not currently being polled because they are not in the scope of any active ping policy.

The following table lists the columns in the unmonitoredIps view.

Table 75. ncmonitor unmonitoredIps view

Column name	Description
ip	A dot-notation IPv4 address that is expected to have been discovered and added to the NCIM topology database.
mainNode	The entityName of the main node, as found in the NCIM entityData table.
mainNodeEntityId	The entityId of the main node, as found in the NCIM entityData table.
entityId	The entityId of the interface, as found in the NCIM entityData table.
class	The entityClass of the main node, as defined by the NCIM classMembers and entityClass tables.
domainMgrId	The ID of the relevant domain, as found in the NCIM topology database domainMgr table.
domainName	The name of the domain identified by the domainMgrId column.

unmanagedIps view

The unmanagedIps view uses the latest poller snapshot from the pollLog table to list any IP addresses from the expectedIps table that are in the scope of active ping policies, but that are not being monitored because they are unmanaged. This is based on the managed status known to the poller at the time of the snapshot.

The following table lists the columns in the unmanagedIps view.

Table 76. ncmonitor unmanagedIps view

Column name	Description
ip	A dot-notation IPv4 address that is expected to have been discovered and added to the NCIM topology database.
mainNode	The entityName of the main node, as found in the NCIM entityData table.
mainNodeEntityId	The entityId of the main node, as found in the NCIM entityData table.
entityId	The entityId of the interface, as found in the NCIM entityData table.
class	The entityClass of the main node, as defined by the NCIM classMembers and entityClass tables.
domainMgrId	The ID of the relevant domain, as found in the NCIM topology database domainMgr table.
domainName	The name of the domain identified by the domainMgrId column.

unpolledFor15MinutesIps view

The unpolledFor15MinutesIps view uses the latest poller snapshot from the pollLog table to list any IP addresses from the expectedIps table that have not been ping polled at all in the last 15 minutes. This includes any IP addresses that are unmanaged or outside the scope of the configured ping polling policies.

The following table lists the columns in the unpolledFor15MinutesIps view.

Table 77. ncmonitor unpolledFor15MinutesIps view

Column name	Description
ip	A dot-notation IPv4 address that is expected to have been discovered and added to the NCIM topology database.
mainNode	The entityName of the main node, as found in the NCIM entityData table.
mainNodeEntityId	The entityId of the main node, as found in the NCIM entityData table.
entityId	The entityId of the interface, as found in the NCIM entityData table.
class	The entityClass of the main node, as defined by the NCIM classMembers and entityClass tables.
domainMgrId	The ID of the relevant domain, as found in the NCIM topology database domainMgr table.
domainName	The name of the domain identified by the domainMgrId column.

delayedPollPolicies view

The delayedPollPolicies view uses the latest poller snapshot from the pollLog table to list all active ping policies are lagging behind schedule.

If the current or previous time interval in the poll cycle (measured from the time at which the poller snapshot was taken) are greater than twice the configured poll interval, the entity and the relevant policy will be listed in this view. This excludes entities that are out of poll scope (see unmonitoredIps view) or that have been unmanaged (see unmanagedIps view), and only applies to the latest snapshot in the pollLog table.

The following table lists the columns in the delayedPollPolicies view.

Table 78. ncmonitor delayedPollPolicies view

Column name	Description
ip	A dot-notation IPv4 address that is expected to have been discovered and added to the NCIM topology database.
mainNode	The entityName of the main node, as found in the NCIM entityData table.
mainNodeEntityId	The entityId of the main node, as found in the NCIM entityData table.
entityId	The entityId of the interface, as found in the NCIM entityData table.

Table 78. ncmonitor delayedPollPolicies view (continued)

Column name	Description
entityType	As defined in the NCIM entityType table, this will be: • 1 for Chassis Ping polls • 2 for Interface Ping polls
policy	The policyName of the ping policy, as found in the NCMONITOR policy table.
configuredPollInterval	The configured pollInterval of the ping policy, as found in the NCMONITOR policy table.
lastPollInterval	The duration of the last complete polling cycle, in seconds.
timeSinceLastPoll	The number of seconds since the last poll, at the time the snapshot was taken.
domainMgrId	The ID of the relevant domain, as found in the NCIM topology database domainMgr table.
domainName	The name of the domain identified by the domainMgrId column.

discoveredIps view

The discoveredIps view lists all IP addresses in the NCIM topology database, together with details of the associated device.

The following table lists the columns in the discoveredIps view.

Table 79. ncmonitor discoveredIps view

Column name	Description
ip	A dot-notation IPv4 address that is expected to have been discovered and added to the NCIM topology database.
mainNode	The entityName of the main node, as found in the NCIM entityData table.
mainNodeEntityId	The entityId of the main node, as found in the NCIM entityData table.
entityId	The entityId of the interface, as found in the NCIM entityData table.
class	The entityClass of the main node, as defined by the NCIM classMembers and entityClass tables.
mainNodeMgdStatus	The current managed status of the main node, as found in the NCIM managedStatus table.
entityMgdStatus	The current managed status of the entity, as found in the NCIM managedStatus table.
domainMgrId	The ID of the relevant domain, as found in the NCIM topology database domainMgr table.
domainName	The name of the domain identified by the domainMgrId column.

managementInterfaceIps view

The managementInterfaceIps view lists SNMP management interface IP addresses for all devices for which Network Manager obtained SNMP access. It does not list the IP addresses of chassis for which no SNMP access was obtained.

For SNMP-accessible devices, the IP address assigned to the chassis by Network Manager must also be the IP address of an interface on that device. This view therefore displays the set of IP addresses which can be monitored by both chassis ping polls and interface ping polls.

The following table describes the columns in the managementInterfaceIps view.

Table 80. ncmonitor managementInterfaceIps view

Column name	Description
ip	A dot-notation IPv4 address that is expected to have been discovered and added to the NCIM topology database.
mainNode	The entityName of the main node, as found in the NCIM entityData table.
mainNodeEntityId	The entityId of the main node, as found in the NCIM entityData table.
entityId	The entityId of the interface, as found in the NCIM entityData table.
class	The entityClass of the main node, as defined by the NCIM classMembers and entityClass tables.
ifIndex	The ifIndex of the interface from the NCIM interface table.
mainNodeMgdStatus	Managed status of the chassis entity identified by the mainNodeEntityId column from the last poller snapshot.
interfaceMgdStatus	Managed status of the interface entity identified by the mainNodeEntityId column from the last poller snapshot.
domainMgrId	The ID of the relevant domain, as found in the NCIM topology database domainMgr table.
domainName	The name of the domain identified by the domainMgrId column.

chassisOnlyIps view

The chassisOnlyIps view lists the IP addresses which can only be monitored with the chassis ping polls, as no interfaces with IP addresses have been discovered on these devices. This is usually the case when Network Manager failed to obtain SNMP access to the device, although it can also depend on the discovery configuration.

The following table lists the columns in the chassisOnlyIps view.

Table 81. ncmonitor chassisOnlyIps view

Column name	Description
ip	A dot-notation IPv4 address that is expected to have been discovered and added to the NCIM topology database.
mainNode	The entityName of the main node, as found in the NCIM entityData table.

Table 81. ncmonitor chassisOnlyIps view (continued)

Column name	Description
mainNodeEntityId	The entityId of the main node, as found in the NCIM entityData table.
class	The entityClass of the main node, as defined by the NCIM classMembers and entityClass tables.
mainNodeMgdStatus	Managed status of the chassis entity identified by the mainNodeEntityId column from the last poller snapshot.
domainMgrId	The ID of the relevant domain, as found in the NCIM topology database domainMgr table.
domainName	The name of the domain identified by the domainMgrId column.

unpollableIps view

The unpollableIps view lists the IP addresses that the poller will not attempt to ping poll. These IP addresses can be monitored using SNMP poll policies.

These are only the secondary IP addresses of multinet interfaces, where a single interface has multiple IP addresses. Ping polls work from the accessIPAddress field of the NCIM chassis and interface tables, and thus only a single IP address per interface can be monitored using ping polls.

The following table lists the columns in the unpollableIps view.

Table 82. ncmonitor unpollableIps view

Column name	Description
ip	A dot-notation IPv4 address that is expected to have been discovered and added to the NCIM topology database.
mainNode	The entityName of the main node, as found in the NCIM entityData table.
mainNodeEntityId	The entityId of the main node, as found in the NCIM entityData table.
class	The entityClass of the main node, as defined by the NCIM classMembers and entityClass tables.
ifIndex	ifIndex The ifIndex of the interface from the NCIM interface table.
interfaceAccessIp	interfaceAccessIp The primary, pollable IP address for the multi-net interface that has this ip as a secondary IP address.
mainNodeMgdStatus	Managed status of the chassis entity identified by the mainNodeEntityId column from the last poller snapshot.
interfaceMgdStatus	interfaceMgdStatus Managed status of the interface entity identified by mainNodeEntityId from the last poller snapshot.
domainMgrId	The ID of the relevant domain, as found in the NCIM topology database domainMgr table.
domainName	The name of the domain identified by the domainMgrId column.

OQL databases

The embedded OQL polling databases provide a number of polling configuration options.

config database for polling

The config database is used by the polling engine, ncp_poller, for a variety of purposes, including diagnostic and debugging purposes, facilitating failover in high availability deployments, debugging the MIB grapher, and configuring the storage limit for historical performance data.

The config database for polling is defined in `$NCHOME/etc/precision/NcPollerSchema.cfg`.

The config database has the following tables:

- config.properties
- config.failover
- config.realTimeControl
- config.pruning

config.properties table

The config.properties table provides the option to configure a number of polling settings.

You can set the values in the config.properties table by editing the following file: `$NCHOME/etc/precision/NcPollerSchema.cfg`.

The following table describes the columns in the config.properties table.

Table 83. config.properties database table

Column name	Constraints	Data type	Description
PolicyUpdateInterval		Integer	Interval in seconds at which the ncp_poller process scans the ncmonitor database for changes to poll policy configuration. Default is 30 seconds.
ManagedStatusUpdateInterval		Integer	Interval in seconds at which the ncp_poller process scans the NCIM managedStatus table for changes. This is the maximum time the poller should take to react to changes in managed status made in any of the following GUIs: Network Views, Network Hop View, Structure Browser. Default is 30 seconds.
LogAccessCredentials		Boolean integer	Controls whether SNMP access credentials (community strings and passwords) appear in plain text. If this is set to false then these strings are replaced with a fixed length string of asterisks. Default is False.

Table 83. config.properties database table (continued)

Column name	Constraints	Data type	Description
UseGetBulk		Boolean integer	By default, GetBulk is not used. Set this parameter to one of the following values: • 0: Default value. GetBulk is not used by the SNMP Helper. • 1: Set this value to configure the SNMP Helper to use GetBulk requests in place of GetNext requests when SNMPv2 or v3 is used.
DefaultGetBulkMaxReps		Integer	This property defines the number assigned to the `max-repetitions` field in GetBulk requests issued by Network Manager processes. The value 20 is used when the GetBulk request contains a single varbind. If multiple varbinds are included, then the value is adjusted accordingly (divided by the number of varbinds), so that responses always contain a similar number of varbinds.

config.failover table

The config.failover provides the option to configure failover settings for polling.

The following table describes the columns in the config.failover table.

Table 84. config.failover database table

Column name	Constraints	Data type	Description
FailedOver	Not NULL	Boolean integer	Used to facilitate failover in high availability deployments. This value must never be modified. You can check this value to determine whether the system has failed over. This field can take the following values: • 0: poller in the primary domain is actively polling, and the backup poller is on standby • 1: primary poller is not actively polling, and the backup poller has taken over

config.realTimeControl table

The config.realTimeControl provides the option to configure settings for the managing real-time poll policies in the MIB Grapher.

The following table describes the columns in the config.realTimeControl table. This table is used by the MIB Grapher application to maintain real-time poll policies. Although the table is not of general use, it can be used to debug MIB graphs if a problem is encountered.

Table 85. config.realTimeControl database table

Column name	Constraints	Data type	Description
POLICYID	Not NULL Primary key	Integer	If there are any real-time graphs active, a record will exist for each one in this table, corresponding to the poll policy created for each graph and referenced using this POLICYID field.
HEARTBEATCOUNT	Not NULL	Integer	Provides an indication of how long the graph has been active. This value represents the number of times the graph has updated the record.
CHANGETIME		Timestamp	UNIX timestamp indicating the last time a 'heartbeat' was received.

config.pruning table

The config.pruning table provides the option to configure the storage limit for historical performance data. The pruning table is used by the Polling engine to configure the limit for the number of rows in the ncpolldata pollData table.

The following table describes the columns in the config.pruning table.

Table 86. config.pruning database table

Column name	Constraints	Data type	Description
MAXPOLLDATAROWS	Not NULL	Long 64	Defines the maximum number of rows allowed in the ncpolldata.polldata table . Once the number of rows exceeds this limit the older data will be pruned, until close to the limit again. Increasing this number will result in an increase in the data storage size required for historical data. **CAUTION:** Increasing this value can also lead to a degradation in the performance of reports using this data.

profiling database for polling

The profiling database is used by the polling engine, ncp_poller, for a variety of purposes, including the storage of summary information for poll policies and poll definitions, ping and SNMP response statistics, and general profiling statistics.

The profiling database for polling is defined in `$NCHOME/etc/precision/NcPollerSchema.cfg`.

The profiling database has the following tables:

- profiling.policy
- profiling.icmp
- profiling.snmp
- profiling.engine

profiling.policy table

The profiling.policy table provides summary information for poll policies and poll definitions.

The following table describes the columns in the profiling.policy table.

Table 87. profiling.policy database table

Column name	Constraints	Data type	Description
AVGSCOPETIME	Not NULL	Integer	The average time taken to evaluate the scope of each poll (not counting the first poll).
FIRSTSCOPETIME	Not NULL	Integer	Time taken, in CPU clock ticks, for the list of entities in scope for the poll to be evaluated for the first time.
ENTITYCOUNT	Not NULL	Integer	Number of entities being monitored by this poll policy and poll definition combination.
POLICYID	Not NULL Primary key	Integer	Value of the ncmonitor.poll.policyId field.
POLICYNAME	Not NULL	Text	Value of the ncmonitor.policy.policyName field.
SCOPETIME	Not NULL	Integer	The total time taken, in CPU clock ticks, for the list of entities in scope for the poll to be evaluated, excluding the first time.
SCOPECOUNT	Not NULL	Integer	The number of times that the scope of the poll has been evaluated.
TARGETCOUNT	Not NULL	Integer	Number of addresses being polled by this poll policy and poll definition combination.
TEMPLATEID	Not NULL Primary key	Integer	Value of the ncmonitor.poll.templateId field.

profiling.icmp table

The profiling.icmp table stores information on ping response statistics.

The following table describes the columns in the profiling.icmp table.

Table 88. profiling.icmp database table

Column name	Constraints	Data type	Description
IPVERSION	Not NULL	Text	IPv4, IPv6, or all versions.
TIMEOUTS	Not NULL	Integer	Number of ICMP requests for which no replay has been received.
PACKETSIN	Not NULL	Text	Number of ICMP packets received.
ERRORSIN	Not NULL	Integer	Number of ICMP errors received.
PACKETSOUT	Not NULL	Integer	Number of ICMP packets sent.
ERRORSOUT	Not NULL	Integer	Total number of errors encountered when sending ICMP packets.

profiling.snmp table

The profiling.snmp table stores information on SNMP response statistics.

The following table describes the columns in the profiling.snmp table.

Table 89. profiling.snmp database table

Column name	Constraints	Data type	Description
ATTRIBUTESIN	Not NULL	Integer	Total number of SNMP errors received.
BACKOFFS	Not NULL	Integer	Total number of times exponential backoff was initiated.
DROPS	Not NULL	Integer	Total number of packets received that were not processed.
ERRORSOUT	Not NULL	Integer	Total number of tooBig errors received.
GETOPERATIONS	Not NULL	Text	Number of SNMP Get operations performed.
GETBULKSOUT	Not NULL	Integer	Total number of SNMP Get Bulk requests sent.
IPADDR	Not NULL	Text	Management IP address of the target device.
GETNEXTOUT	Not NULL	Integer	Total number of SNMP Get Next requests sent.
GETSOUT	Not NULL	Integer	Total number of SNMP Get requests sent.
NOSUCHNAMESIN	Not NULL	Integer	Total number of noSuchName errors received.
PACKETSIN	Not NULL	Integer	Total number of SNMP packets received, including errors.
PACKETSOUT	Not NULL	Integer	Total number of SNMP packets sent.
RETRIES	Not NULL	Integer	Total number of retries.
SETERRORSIN	Not NULL	Integer	Number of errors received from Set requests.

Table 89. profiling.snmp database table (continued)

Column name	Constraints	Data type	Description
SETOPERATIONS	Not NULL	Integer	Number of SNMP Set operations performed.
SETSOUT	Not NULL	Integer	Total number of Set requests sent.
TIMEOUTS	Not NULL	Integer	Total number of SNMP operations that timed out.
TOOBIGSIN	Not NULL	Integer	Number of tooBig errors received.
WALKOPERATIONS	Not NULL	Integer	Number of SNMP Walk operations performed.

profiling.engine table

The profiling.engine table stores general profiling statistics information.

The following table describes the columns in the profiling.engine table.

Table 90. profiling.engine database table

Column name	Constraints	Data type	Description
STARTTIME	Not NULL	Timestamp	Time when profiling started.
LASTUPDATE	Not NULL	Timestamp	Last time profiling statistics were updated.
THREADSINUSE	Not NULL	Integer	Number of active threads in the core polling engine.
BATCHESQUEUED	Not NULL	Integer	Number of batches that should be running but for which there are no threads available.
AVGBATCHTIME	Not NULL	Integer	Average time in milliseconds to process each batch of work.

Appendix H. Event enrichment databases

Use this information to understand the structure of databases used for event enrichment and for the Event Gateway plug-ins.

ncp_g_event database

The Event Gateway database enables ncp_g_event, the Event Gateway, to transfer data between Network Manager and Tivoli Netcool/OMNIbus.

The ncp_g_event database has the following database schema: config

The default configuration of the gateway is used for most systems. You can make adjustments to the configuration settings by modifying the values inserted into the Event Gateway config database. This database contains the configuration settings that define the operation of the Event Gateway. For example, you can modify the mappings used between Network Manager and Tivoli Netcool/OMNIbus and the filters that determine which events are processed.

Entity data used by the Event Gateway is stored in NCIM cache, which is a copy of the NCIM topology database. For more information on NCIM cache see the *IBM Tivoli Network Manager IP Edition Topology Database Reference*.

For information about ncp_g_event command-line options, see the *IBM Tivoli Network Manager IP Edition Administration Guide*.

The config database schema

The config database is used to configure event mapping between Tivoli Netcool/OMNIbus and Network Manager.

The config database can also be used to define filters that limit the number of events passed between Tivoli Netcool/OMNIbus and Network Manager.

The table below summarizes the config database schema. This schema is defined in `NCHOME/etc/precision/EventGatewaySchema.cfg`. You can specify domain-specific versions of this file using the format: `NCHOME/etc/precision/EventGatewaySchema`*`domain_name`*`.cfg`, where *domain_name* is the name of your domain; for example, `NCHOME/etc/precision/EventGatewaySchemaNCOMS.cfg`.

Table 91. config database summary

Database name	`config`
Defined in	`NCHOME/etc/precision/EventGatewaySchema.cfg`

Fully qualified database table names	`config.defaults`
	`config.eventMaps`
	`config.failover`
	`config.nco2ncp`
	`config.ncp2nco`
	`config.precedence`

© Copyright IBM Corp. 2006, 2016

The topics below describe the database tables of the config database.

config.defaults table

The config.defaults table contains general configuration data for the Event Gateway.

The table below describes the config.defaults table.

Note: The fields `NcoAuthUserName` and `NcoAuthPassword` are now configured in the `$NCHOME/etc/precision/NcoLogins.DOMAIN.cfg file`.

Table 92. config.defaults Table Description

Column Name	Constraints	Data Type	Description
`IDUCFlushTime`	NOT NULL	Integer	Specifies the interval, in seconds, between Insert Delete Update Control (IDUC) flushes from the ObjectServer. The default value is 5.
`ObjectServerUpdate Interval`	NOT NULL	Integer	Specifies the interval that the Event Gateway uses to queue event enrichment updates to the ObjectServer.
`NcpServerEntity`	NOT NULL	Text	Specifies the IP address of the polling station. By default, the gateway assumes that the polling station for Network Manager IP Edition is running on the local host. If you want to set a different polling station, specify the IP address of the polling station in the NcpServerEntity field. **Note:** Root cause analysis (RCA) cannot perform isolated suppression if the device specified in NcpServerEntity is not present and connected within the topology.

Related tasks:

"Configuring the ObjectServer update interval field" on page 157
You can configure the interval that the Event Gateway uses to queue event enrichment updates to the ObjectServer.

"Configuring the poller entity" on page 169
To enable the RCA plugin to perform isolated suppression when the Network Manager server is not within the scope of your network domain, specify the IP address or DNS name of the ingress interface as the poller entity.

config.precedence table

The config.precedence table lists events by event ID and contains the information necessary to determine which event has precedence when multiple events occur on the same interface. Based on the event ID, the config.precedence table also determines which event map to use to process an event from Tivoli Netcool/OMNIbus.

The table below describes the config.precedence table.

Table 93. config.precedence Table Description

Column Name	Constraints	Data Type	Description
`Precedence`	NOT NULL	Integer	Specifies the number used by the root-cause analysis (RCA) plug-in when there are multiple events on the same entity within the network topology. The number is used to determine which of the events has precedence and therefore suppresses the other event on the interface. If a link down event has a higher Precedence value than a ping fail event, then the link down event suppresses the ping fail event on the interface. These Precedence values are unique. • 0 - An event with this Precedence value cannot suppress any other events. The event cannot become a root cause event. If the Precedence value is set to 0, then the event can become a symptom event or be marked as cause unknown. • 10000 and greater - An event with a Precedence value greater than or equal to 10000 cannot be suppressed; and the event cannot become a symptom event. The event can only become a root cause event or be marked as cause unknown.
`EventMapName`	NOT NULL	Text	Specifies the name of the event map from the config.eventMaps table that is used to process the event with a matching EventId.
`NcoEventId`	PRIMARY KEY NOT NULL	Text	Specifies the mapping from the EventId in the alerts.status table to the values of Precedence and EventMapName defined in this table. **Note:** If an event is not listed in this table, then the event is handled by the generic-ip event map.

Related concepts:

"Precedence value" on page 131
At the same time that an event map is selected to handle the event, a numerical precedence value is associated with an event. This precedence value is used by the RCA plugin in cases where there are multiple events on the same entity. The event with the highest precedence value on the entity is used to suppress other events.

config.eventMaps Table

The config.eventMaps table contains the event map that specifies how an event is processed. The table holds information specific to each type of Tivoli Netcool/OMNIbus event that is processed by the Event Gateway.

The table below describes the config.eventMaps table.

Table 94. config.eventMaps table description

Column Name	Constraints	Data Type	Description
`EventMapName`	PRIMARY KEY NOT NULL	Text	Specifies the name of the event map. This value is referenced by the config.precedence table.
`HandledBy`		Text	An alternative to the PolledEntityStitcher, and provided for backwards compatibility. Some legacy gateway eventMaps are redundant, but removing them completely would create upgrade problems. This field allows a legacy eventMap to be mapped to a new eventMap, and behave as if the event had been handled by that eventMap.
`EventCanFlap`		Boolean	Indicates if it is possible for the event to flap. Flapping is a condition where a device or interface connects to and then disconnects from the network repeatedly in a short space of time. This causes problem and clear events to be received one after the other for the same device or interface. Setting the `EventCanFlap = 1` informs the RCA plug-in of this condition. The RCA plug-in places these events in the mojo.events database with `TimedEscalation = 1` and are left there for 30 seconds. After 30 seconds one of the RCA plug-in stitchers processes all events that are at least 30 seconds old and have the `TimedEscalation = 1` setting. By waiting 30 seconds to process the event, the system ensures that the entity that generated the event has settled down and is not flapping.

Related concepts:

"Stitcher selection using event maps" on page 102
Use this information to understand how the Event Gateway is configured to enable event maps to call specific Event Gateway stitchers.

config.nco2ncp table

The config.nco2ncp table is used to filter events being passed from Tivoli Netcool/OMNIbus to Network Manager.

The table below describes the config.nco2ncp table.

Table 95. config.nco2ncp table description

Column Name	Constraints	Data Type	Description
`EventFilter`	NOT NULL	Text	Specifies a filter that indicates which events should be processed by the Event Gateway. Events that match the filter are processed. **Attention:** Do not modify this filter unless you are aware of the consequences of the modification. Only advanced users should modify this filter.
`StandbyEvent Filter`		Text	Used when the primary server is down and the backup server is active. The standby filter only allows ItnmHealthCheck events through the Event Gateway. These events are passed to the Failover plugin and tell the system to switch back to primary mode.
`FieldFilter`	Externally defined vblist data type	Object	Specifies a subset of alerts.status fields that are passed through to the Event Gateway. If the field filter is empty then all alerts.status fields are are passed through. The purpose of this filter is to limit the fields passed through to the minimum required set in order to lighten the processing load.

The gateway determines whether to insert a new record or update an existing record according to whether the ObjectServer sends the event as an insert using IDUC or as an update.

Related concepts:

"Incoming event filter" on page 86
The incoming event filter filters out events from the ObjectServer and only passes events that meet certain criteria.

"Standby filter" on page 88
In a failover deployment, the standby filter is used by the backup domain in a failover pair. That means the backup domain when the primary is active, or the primary domain if the backup is active. The standby filter only allows health check (ItnmHealthCheck) events through the Event Gateway. These events are passed to the Failover plugin and tell the system to switch back to primary mode. Note that for failover behaviour, any modifications to this filter must still ensure that the standby filter accepts health check events.

config.ncp2nco table

The config.ncp2nco table is used to filter and map events passed from Network Manager IP Edition to Tivoli Netcool/OMNIbus.

The table below describes the config.ncp2nco table.

Table 96. config.ncp2nco table description

Column Name	Constraints	Data Type	Description
FieldFilter	Externally defined vblist data type	Object	Specifies the set of ObjectServer fields that may be updated by the Event Gateway.

Related concepts:

"Outgoing field filter" on page 90
The outgoing field filter defines the set of ObjectServer fields that may be updated by the Event Gateway.

config.failover table

The config.failover table contains the failover configuration and current failover state of the Event Gateway component.

Attention: Do not manually change the values of the config.failover table. In a failover configuration, the FailedOver field is modified by the virtual domain process.

The table below describes the config.failover table.

Table 97. config.failover table description

Column Name	Constraints	Data Type	Description
`FailedOver`	NOT NULL	Boolean	Specifies the failover state. • 0 - Not in a failover state • 1 - In a failover state

ncp_g_event plug-in databases

The Event Gateway plug-in database tables are used by the plug-ins to store processing data.

RCA plug-in database

The RCA plug-in database tables enable the RCA plug-in to perform root-cause analysis.

mojo.events events database table

The `mojo` database stores all event records sent for root cause analysis by the Event Gateway. The database contains the mojo.events table.

The `mojo` database is defined in `NCHOME/etc/precision/RCASchema.cfg`.

The column names of the records are used in many of the conditional filters when constructing event correlation methods.

The table below describes columns in the mojo.events table.

Table 98. Descriptions for the mojo.events database table columns

Column Name	Constraints	Data type	Description
ChangeTime	TIMESTAMP Not null	Long Integer	Specifies the time the event was last updated by the RCA plug-in.
CreateTime	TIMESTAMP Not null	Long Integer	Specifies the time the event was first seen by the RCA plug-in.
Description		Text	Specifies a textual description of the event.
EntityType	Not null	Int	A value of 1 or 8 indicates that this is a chassis device.
EventId		Text	Type of event; for example NmosPingFail.
FirstOccurrence	TIMESTAMP Not null		Time the event was first seen by Tivoli Netcool/OMNIbus. **Note:** This value is set by the probe, not by Tivoli Netcool/OMNIbus. This means that this field arrives at the ObjectServer with a value already set. Tivoli Netcool/OMNIbus never touches this field.
IsIsolationPoint	Not null	Int	Can take the following values: • 0 - No • 1 - Yes
IsLoopbackInterface	Not null	Int	Can take the following values: • 0 - No • 1 - Yes
IsMasterEvent	Not null	Int	Can take the following values: • 0 - This is *not* the master event on the entity. • 1 - This is the master event on the entity. **Note:** The master event on an entity will suppress all other events on that entity, should there be any.
IsOrphan	Not null	Int	Can take the following values: • 0 - No • 1 - Yes **Note:** This field is used internally to enable the RCA plug-in to reprocess suppressed events whose root cause event has since been deleted.

Table 98. Descriptions for the mojo.events database table columns (continued)

Column Name	Constraints	Data type	Description
LastOccurrence	TIMESTAMP Not null	Long Integer	Time the event was last seen by Tivoli Netcool/OMNIbus. **Note:** This value is set by Tivoli Netcool/OMNIbus itself when it receives the event.
NmosCauseType	Not null	Int	Can take the following values: • 0 - Unknown • 1 - Root cause • 2 - Symptom • 3 - Not suppressing and not suppressed
NmosEntityId	Not null	Int	Entity on which the event occurred.
NmosObjInst	Not null	Int	Entity ID for the chassis related to the entity on which the event occurred.
NmosSerial	Not null	Int	Serial number of the event that suppressed this event.
Precedence		Int	A value from 0 to 10,000 indicating, where there are multiple events on the same entity, the event to be used to suppress the other events on that entity. The event with the highest precedence value suppresses the others.
RemoteNodeAlias		Text	Network address of the remote network entity
Serial	Primary key Not null	Uint	Serial number of this event in Tivoli Netcool/OMNIbus. Used to uniquely identify the event and the record in mojo.events.
Severity	Not null	Int	Severity of the event.
State	Not null	Int	Event state for this event.
SuppressionState	Not null	Int	Suppression state for this event. This field can take the following values: • 0 - No suppression • 1 - Entity suppression • 2 - Contained suppression • 3 - ConnectedSuppression • 4 - IsolatedSuppression • 5 - PeerSuppression
SuppressionTime	TIMESTAMP Not null	Long Integer	Time the event was last suppressed.
TimedEscalation	Not null	Int	Can take the following values: • 0 - Tells RCA plug-in to process the event immediately. • 1 - Tells RCA plug-in to process the event after 30 seconds. This is usually set by events that can flap. • 2 - Set for events that previously had the `TimedEscalation = 1` setting and have since been processed.

config.defaults database table

The config.defaults database table stores configuration data for the RCA plug-in event queue.

The `config` database is defined in `NCHOME/etc/precision/RCASchema.cfg`.

The table below describes columns in the config.defaults database table.

Table 99. Descriptions for the config.defaults database table columns

Column Name	Constraints	Data type	Description
RequeueableEventIds		Text	Specifies that events of certain types can be requeued if the RCA plug-in queue becomes very large. This ensures that only one event of a specific event type exists in the queue at any one time.
MaxAgeDifference	NOT NULL	Text	Specifies the maximum age difference between events that pass through the RCA plug-in. Events that have a difference in age greater than this specified value cannot suppress each other. By default this option is switched off, that is, set to 0. This means that events on the same entity suppress each other regardless of the age of the events.
HonourManagedStatus	Integer	Boolean	Specifies whether the RCA plugin uses the managed status of an entity in calculating the root cause of an event. If this value is set to 1, then events from unmanaged devices are ignored.

Related concepts:

"RCA and unmanaged status" on page 134
Use this information to understand how the RCA plug-in handles events from devices that are in unmanaged state, also known as maintenance state.

Related tasks:

"Configuring the maximum age difference for events" on page 170
By default, events on the same entity suppress each other regardless of the age of the events. An event received today can suppress an event received yesterday on the same entity. You can change this by specifying a maximum age difference between events that pass through the RCA plug-in. Events that have a difference in age greater than this specified value cannot suppress each other.

SAE plug-in database

The SAE plug-in database tables enable the SAE plug-into generate service-affected events for services such as MPLS VPNs and IP paths.

The table below summarizes the config database schema.

Table 100. config database summary

Database name	config

Table 100. config database summary (continued)

Defined in	NCHOME/etc/precision/SaeSchema.cfg NCHOME/etc/precision/SaeIPPath.cfg NCHOME/etc/precision/SaeItnmService.cfg NCHOME/etc/precision/SaeMplsVpn.cfg
Fully qualified database table names	config.serviceTypes

Related concepts:

"SAE plug-in" on page 125
The SAE plug-in generates service-affected events for MPLS VPNs and IP paths.

config.serviceTypes table

The config.serviceTypes table contains configuration information for the SAE plug-in.

The table below describes columns in the config.serviceTypes table.

Table 101. Descriptions for the config.serviceTypes database table columns

Column Name	Constraints	Data type	Description
ServiceTypeName	Primary key Not null	Text	Represents the type of service; for example, "MPLS VPN Edge Service" or "IP Path". This string will appear in the eventId field of the SAE event in the ObjectServer and will also form part of the Summary field of the SAE event in the ObjectServer.
CollectionEntityType	Not null	Integer	Used to specify the entity type that corresponds to the collection that the SAE will be generated for; for example 17 (VPN), 34 (ITNM Service), or 80 (IP Path). For a listing of possible entity types in the NCIM entityType table, see the *IBM Tivoli Network Manager Topology Database Guide*.
ConstraintFilter	Optional	Text	Used to constrain the entries of interest in the collection table, if necessary. For example, for the table networkVpn, entries that have Type = 'MPLS Core' are excluded. In this case, the contraint filter is formulated as follows: "networkVpn->VPNTYPE <> 'MPLS Core'"
CustomerNameField	Optional	Text	Used to specify where to obtain the customer name string to append to the Summary field of the event in the ObjectServer. For example, if the ServiceEntity record contains the field Customer then this can be used to retrieve a string such as "ACME Inc" from the ServiceEntity topology record. This example would take the form "entityData->DESCRIPTION".

ncp_g_event plug-in database tables in ncmonitor

Use this information to understand which Event Gateway configuration tables are available in the ncmonitor database and what type of information each table contains. Most of these tables relate to Event Gateway plug-in configuration.

The table below lists each Event Gateway plug-in configuration tables in the ncmonitor database and explains the purpose of the table.

Table 102. Event Gateway plug-in configuration tables in ncmonitor

Table	Description
ncmonitor. gwPluginTypes	Lists the available plugin libraries. Similar idea to poller templates/poll definitions. This should contain a single entry for the Adaptive Polling plugin.
ncmonitor. gwPlugins	Lists the plugins that are enabled. Similar idea to the poller policies. This should contain a handful of entries for the Adaptive Polling plugin.
ncmonitor. gwPluginEventMaps	Identifies the event maps that each plugin is interested in. Plugins are only supplied with events handled by listed event maps.
ncmonitor. gwPluginEventStates	Identifies the type of event that each plugin is interested in. Plugins are only supplied with events handled by listed event states.
ncmonitor. gwSchemaFiles	Lists the OQL schema files to be read by the Event Gateway. These files are read in the order they are listed, and therefore the EventGatewaySchema file should be the first file listed, as it defines the tables. By default, this lists the EventGatewaySchema and NcoGateInserts files. The purpose of this table is to allow additional self-contained schema files to be supplied, with the same format as the existing NcoGateInserts file, for future device support.
ncmonitor. gwPluginConf	Allows optional configuration variables to be defined for specific plugins. **Note:** It is intended for values that would not, under normal circumstances, be changed.

Related concepts:

"Adaptive polling plug-in" on page 119
Use this information to understand plug-in prerequisites, how the adaptive polling plug-in populates fields in the activeEvent table, as well as configuration details associated with the plug-in. The activeEvent table is in the NCMONITOR schema.

"Disco plug-in" on page 122
Use this information to understand some basic information about how this plug-in operates, plug-in prerequisites, and configuration details associated with the plug-in.

"zNetView plug-in" on page 126
Use this information to understand plug-in prerequisites as well as configuration details associated with the plug-in.

Related tasks:

"Enabling and disabling plug-ins" on page 161
Use the `ncp_gwplugins.pl` script to enable and disable plug-ins. Run the script separately for each plug-in.

"Listing plug-in information" on page 162
You can list information on Event Gateway plug-ins. For example, you can list the event maps and event states that each plug-in subscribes to.

"Modifying event map subscriptions" on page 163
You can change the event maps that a plug-in subscribes to. For example, if you add a new event map and want the system to perform RCA on events handled by that event map, then you must add that event map to the subscription list for the RCA plug-in.

"Setting plug-in configuration parameters" on page 165
You can set optional configuration parameters for the Event Gateway plug-ins using the `ncp_gwplugins.pl` script.

Appendix I. Network Manager glossary

Use this information to understand terminology relevant to the Network Manager product.

The following list provides explanations for Network Manager terminology.

AOC files
Files used by the Active Object Class manager, `ncp_class` to classify network devices following a discovery. Device classification is defined in AOC files by using a set of filters on the object ID and other device MIB parameters.

active object class (AOC)
An element in the predefined hierarchical topology of network devices used by the Active Object Class manager, `ncp_class`, to classify discovered devices following a discovery.

agent See, discovery agent.

class hierarchy
Predefined hierarchical topology of network devices used by the Active Object Class manager, `ncp_class`, to classify discovered devices following a discovery.

configuration files
Each Network Manager process has one or more configuration files used to control process behaviour by setting values in the process databases. Configuration files can also be made domain-specific.

discovery agent
Piece of code that runs during a discovery and retrieves detailed information from discovered devices.

Discovery Configuration GUI
GUI used to configure discovery parameters.

Discovery engine (ncp_disco)
Network Manager process that performs network discovery.

discovery phase
A network discovery is divided into four phases: Interrogating devices, Resolving addresses, Downloading connections, and Correlating connectivity.

discovery seed
One or more devices from which the discovery starts.

discovery scope
The boundaries of a discovery, expressed as one or more subnets and netmasks.

Discovery Status GUI
GUI used to launch and monitor a running discovery.

discovery stitcher
Piece of code used during the discovery process. There are various discovery stitchers, and they can be grouped into two types: data collection stitchers, which transfer data between databases during the data collection

© Copyright IBM Corp. 2006, 2016

phases of a discovery, and data processing stitchers, which build the network topology during the data processing phase.

domain
See, network domain.

entity A topology database concept. All devices and device components discovered by Network Manager are entities. Also device collections such as VPNs and VLANs, as well as pieces of topology that form a complex connection, are entities.

event enrichment
The process of adding topology information to the event.

Event Gateway (ncp_g_event)
Network Manager process that performs event enrichment.

Event Gateway stitcher
Stitchers that perform topology lookup as part of the event enrichment process.

failover
In your Network Manager environment, a failover architecture can be used to configure your system for high availability, minimizing the impact of computer or network failure.

Failover plug-in
Receives Network Manager health check events from the Event Gateway and passes these events to the Virtual Domain process, which decides whether or not to initiate failover based on the event.

Fault Finding View
Composite GUI view consisting of an **Active Event List (AEL)** portlet above and a Network Hop View portlet below. Use the Fault Finding View to monitor network events.

full discovery
A discovery run with a large scope, intended to discover all of the network devices that you want to manage. Full discoveries are usually just called discoveries, unless they are being contrasted with partial discoveries. See also, partial discovery.

message broker
Component that manages communication between Network Manager processes. The message broker used byNetwork Manager is called Really Small Message Broker. To ensure correct operation of Network Manager, Really Small Message Broker must be running at all times.

NCIM database
Relational database that stores topology data, as well as administrative data such as data associated with poll policies and definitions, and performance data from devices.

ncp_disco
See, Discovery engine.

ncp_g_event
See, Event Gateway.

ncp_model
See, Topology manager.

ncp_poller
See, Polling engine.

network domain
A collection of network entities to be discovered and managed. A single Network Manager installation can manage multiple network domains.

Network Health View
Composite GUI view consisting of a Network Views portlet above and an **Active Event List (AEL)** portlet below. Use the Network Health View to display events on network devices.

Network Hop View
Network visualization GUI. Use the Network Hop View to search the network for a specific device and display a specified network device. You can also use the Network Hop View as a starting point for network troubleshooting. Formerly known as the Hop View.

Network Polling GUI
Administrator GUI. Enables definition of poll policies and poll definitions.

Network Views
Network visualization GUI that shows hierarchically organized views of a discovered network. Use the Network Views to view the results of a discovery and to troubleshoot network problems.

OQL databases
Network Manager processes store configuration, management and operational information in OQL databases.

OQL language
Version of the Structured Query Language (SQL) that has been designed for use in Network Manager. Network Manager processes create and interact with their databases using OQL.

partial discovery
A subsequent rediscovery of a section of the previously discovered network. The section of the network is usually defined using a discovery scope consisting of either an address range, a single device, or a group of devices. A partial discovery relies on the results of the last full discovery, and can only be run if the Discovery engine, `ncp_disco`, has not been stopped since the last full discovery. See also, full discovery.

Path Views
Network visualization GUI that displays devices and links that make up a network path between two selected devices. Create new path views or change existing path views to help network operators visualize network paths.

performance data
Performance data can be gathered using performance reports. These reports allow you to view any historical performance data that has been collected by the monitoring system for diagnostic purposes.

Polling engine (ncp_poller)
Network Manager process that polls target devices and interfaces. The Polling engine also collects performance data from polled devices.

poll definition
Defines how to poll a network device or interface and further filter the target devices or interfaces.

poll policy
Defines which devices to poll. Also defines other attributes of a poll such as poll frequency.

Probe for Tivoli Netcool/OMNIbus (nco_p_ncpmonitor)
Acquires and processes the events that are generated by Network Manager polls and processes, and forwards these events to the ObjectServer.

RCA plug-in
Based on data in the event and based on the discovered topology, attempts to identify events that are caused by or cause other events using rules coded in RCA stitchers.

RCA stitcher
Stitchers that process a trigger event as it passes through the RCA plug-in.

root-cause analysis (RCA)
The process of determining the root cause of one or more device alerts.

SNMP MIB Browser
GUI that retrieves MIB variable information from network devices to support diagnosis of network problems.

SNMP MIB Grapher
GUI that displays a real-time graph of MIB variables for a device and usse the graph for fault analysis and resolution of network problems.

stitcher
Code used in the following processes: discovery, event enrichment, and root-cause analysis. See also, discovery stitcher, Event Gateway stitcher, and RCA stitcher.

Structure Browser
GUI that enables you to investigate the health of device components in order to isolate faults within a network device.

Topology Manager (ncp_model)
Stores the topology data following a discovery and sends the topology data to the NCIM topology database where it can be queried using SQL.

WebTools
Specialized data retrieval tools that retrieve data from network devices and can be launched from the network visualization GUIs, Network Views and Network Hop View, or by specifying a URL in a web browser.

Notices

This information applies to the PDF documentation set for IBM Tivoli Network Manager IP Edition 3.9.

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web

© Copyright IBM Corp. 2006, 2016

sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
958/NH04
IBM Centre, St Leonards
601 Pacific Hwy
St Leonards, NSW, 2069
Australia
IBM Corporation
896471/H128B
76 Upper Ground
London
SE1 9PZ
United Kingdom
IBM Corporation
JBF1/SOM1 294
Route 100
Somers, NY, 10589-0100
United States of America

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the

names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Trademarks

The terms in Table 103 are trademarks of International Business Machines Corporation in the United States, other countries, or both:

Table 103. IBM trademarks

AIX	iSeries	RDN
ClearQuest	Lotus	SecureWay
Cognos	Netcool	solidDB
Current	NetView	System z
DB2	Notes	Tivoli
developerWorks	OMEGAMON	WebSphere
Enterprise Storage Server	PowerVM	z/OS
IBM	PR/SM	z/VM
Informix	pSeries	zSeries

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Privacy policy considerations

IBM Software products, including software as a service solutions, ("Software Offerings") may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering's use of cookies is set forth below.

This Software Offering does not use cookies or other technologies to collect personally identifiable information.

For more information about the use of various technologies, including cookies, for these purposes, see IBM's privacy policy at http://www.ibm.com/privacy.

Index

Special characters

<$nopage>root cause analysis, see RCA
  examples 139
  poller entity 140

A

accessibility x
adaptive polling
  managing 51
adaptive polling plug-in 119
adaptive polling scenario
  confirmation of threshold violation 54
  device down confirmation 51
adaptive polling scenarios 51
adaptive polls
  creating 56
adding SAE types to SAE plug-in 167
administering
  historical polled data 70
  multiple poller feature 65
  network polling 59
  poll policy throttling 62
  polls 59
alerts.status table
  fields used for Network Manager 201
AMOS process
  root cause analysis 141
audience v

B

basic threshold expression
  example 44
basic threshold policies
  default 174
basic threshold poll definitions
  changing 36
  creation 23
basic threshold polls
  changing 31

C

cards
  entities 143
changing
  basic threshold poll definitions 36
  chassis ping poll definitions 40
  generic threshold poll definitions 38
  interface ping poll definitions 40
  interval for checking poll policy membership size 64
  link state poll definitions 41
  poll definitions 36
  poll policies 31
  remote ping poll definitions 41
changing poll definitions
  example 43
chassis devices
  downstream 144
chassis ping poll definitions
  changing 40
  creating 27
chassisOnlyIps view 223
checking topology paths used by RCA 148
Chinese characters
  See multibyte data
clear threshold
  example 187
commands
  SIGHUP 158
config database 231
  for polling 225
config.defaults database 232
  root cause analysis 239
config.eventMaps 233
config.eventMaps database 233
config.failover database 236
config.failover table
  for polling 226
config.nco2ncp database 234
config.ncp2nco database 236
config.ncp2nco table 236
config.precedence database 232
config.properties table
  for polling 225
config.pruning table
  for polling 227
config.realTimeControl table
  for polling 227
config.serviceTypes table 240
configuration parameters for Event Gateway plug-ins
  setting 165
configuring
  event enrichment 153
  Event Gateway plug-ins 161
  extra event enrichment 153
  poller entity 169
  RCA plug-in 169
  root-cause analysis 169
  SAE plug-in 166
connected interfaces 142
connecting entities 143
contained interfaces 141
conventions, typeface xi
copying
  get_policies.pl program 61
  polls
    across domains 61
creating
  adaptive polls 56
  basic threshold poll definitions 23
  chassis ping poll definitions 27
  generic threshold poll definitions 25
  interface ping poll definitions 27
creating *(continued)*
  link state poll definitions 29
  poll definitions 23
  remote ping poll definitions 29

D

data extraction stitchers 109
data labels 9
database tables
  RCA plug-in 236
  SAE plug-in 239
databases 233, 236
  config 231
  config.defaults 232, 239
  config.failover 236
  config.nco2ncp 234
  config.precedence 232
  for event enrichment 231
  for polling 213
  gateway 231
  mojo.events 237
  ncmonitor 213
  ncp_g_event 231
databases for polling
  config.failover table 226
  config.properties table 225
  config.pruning table 227
  config.realTimeControl table 227
  profiling.engine table 230
  profiling.icmp table 229
  profiling.policy table 228
  profiling.snmp table 229
default event maps 99
default precedence values 132
defining using multibyte data 10
delayedPollPolicies view 221
deleting
  poll definitions 49
  poll policies 47
descriptions
  RCA stitchers 138
devices
  chassis 142, 144
  Chassis 141
  main node 141
disabling
  Event Gateway plug-ins 161
  network view updates for poll policies 64
  poll policies 60
disabling polls 11
disco plug-in 122
discoveredIps view 222
downstream 140
downstream chassis devices
  root cause analysis 144
downstream devices
  root cause analysis 139

© Copyright IBM Corp. 2006, 2016

E

education
  see Tivoli technical training x
enabling
  Event Gateway plug-ins 161
  network view updates for poll policies 64
  poll policies 60
enabling polls 11
entities
  cards 143
  virtual routers 143
  VLANs 143
entity retrieval stitchers 110
EntityFromIfString.stch 111
environment variables, notation xi
eval statement
  in threshold expressions 189
event categories 207
event correlation 83
event enrichment 83
  assigning states to different event types 93
  configuring 153
  configuring extra event enrichment 153
  data extraction stitchers 109
  enriching event with interface name 155
  enriching event with main node device location 154
  entity retrieval stitchers 110
  event enrichment stitchers 112
  event filter 86
  event filtering 85
  event handling 97
  event maps 97
  event state 93
  event state diagram 94
  example 115, 154, 155
  field filter 89, 90
  incoming event filter 86
  incoming field filter 89
  outgoing field filter 90
  poller entity 140
  precedence value 131
  quick reference 83
  standby filter 88
  stitchers 105
  stitchers not used by default 114
  topology lookup stitchers 105
event enrichment databases 231
event enrichment stitchers 112
event fields 199
event filter 86
  incoming to Event Gateway 85
event filtering 85
Event Gateway
  assigning states to different event types 93
  configuring 158
  configuring extra event enrichment 153
  data extraction stitchers 109
  entity retrieval stitchers 110
  event enrichment stitchers 112
  event handling 97
Event Gateway *(continued)*
  event maps 97
  event state 93
  event state diagram 94
  filter 85
  incoming filter 85
  logging into the databases using OQL 158
  methods for selecting event maps 98
  NcpServerEntity 133
  outgoing queue 91
  plug-in configuration tables 241
  poller entity 140
  Poller entity 133
  precedence value 131
  precedence values 132
  selecting event maps 97
  standby filter 88
  stitchers 102
  stitchers not used by default 114
  topology lookup stitchers 105
Event Gateway plug-in configuration parameters
  setting 165
Event Gateway plug-in information
  listing 162
Event Gateway plug-ins
  configuring 161
  disabling 161
  enabling 161
  subscriptions 163
Event Gateway plugins 118
  RCA 130
Event Gateway SAE plug-in
  adding SAE types 167
Event Gateway stitchers
  example 107, 109, 111, 113
event handling 97
event map
  selection 97
  selection methods 98
  selection using the Event Gateway 97
event mappings 233, 236
event maps 97
  default 99
event state 93
event state diagram 94
event types 93
events
  maximum age difference for RCA 170
  network 208
  service-affected (SAE) summary fields 166
  status information 208
example
  data extraction stitcher 109
  entity retrieval stitcher 111
  event enrichment 115, 154, 155
  event enrichment stitcher 113
  Event Gateway stitcher 107, 109, 111, 113
  poll definition example of threshold expression 44
  threshold expression 44
example *(continued)*
  topology lookup stitcher 107
examples
  threshold polling 7
expectedIps table 216
extra event enrichment
  configuring 153
ExtractIfString.stch 109

F

failover plug-in 123
failures
  chassis 142
field filter 89, 90
field mappings
  Network Manager to alerts.status 201
filter
  incoming to Event Gateway 85

G

generic threshold expression
  example 44
generic threshold policies
  default 174
generic threshold poll definitions
  changing 38
  creation 25
generic threshold polls
  changing 31
  examples 187
glossary 243

H

historical polled data
  administering 70
historical polling
  deleting data 73

I

incoming event filter 86
incoming field filter 89
interface ping poll definitions
  changing 40
  creating 27
interfaces 145
  contained 141
  downstream 142
  loopback 141
  upstream 142
interval for checking poll policy membership size
  changing 64
isolated suppression 144, 147

L

link state poll definitions
  changing 41
  creating 29

link state polls
changing 31
listing
Event Gateway plug-in information 162
locales
See multibyte data
lookup stitchers 105
LookupIp.stch 107
loopback interfaces 141

M

managed status
and RCA 134
managementInterfaceIps view 223
managing
adaptive polling 51
manuals vii
maximum age differencefor RCA 170
membership size of poll policies
changing checking interval 64
mojo.events database
root cause analysis 237
multibyte data
in poll definitions 10
multiple poller feature
adding poller 67
administering 65
overview 66
removing poller 69

N

NCIM database
querying using OQL 158
NCMONITOR
polling databases 213
polling status tables 216
ncmonitor database
chassisOnlyIps view 223
delayedPollPolicies view 221
discoveredIps view 222
Event Gateway plug-in configuration tables 241
expectedIps table 216
managementInterfaceIps view 223
pollLog table 216
pollLogSummary table 218
SNMP 213
snmpAccess table 214
snmpTarget table 213
snmpUser table 215
snmpv1Sec table 215
snmpv3Sec table 215
undiscoveredIps view 219
unmanagedIps view 220
unmonitoredIps view 219
unpollableIps view 224
unpolledFor15MinutesIps view 221
ncp_g_event 231
NcpServerEntity 133
network events 208
Network Manager event categories 207
Network Manager event fields 199
Network Manager glossary 243
Network Manager to alerts.status mappings 201
network polling
administering 59
network view updates for poll policies
disabling 64
enabling 64

O

ObjectServer
querying using OQL 158
online publications vii
operators
in threshold expressions 192
OQL polling databases 225
ordering publications vii
outgoing field filter 90
outgoing queue
on Event Gateway 91
overview
multiple poller feature 66

P

parameters
of poll definitions 4
of poll policies 2
pathsfor RCA 148
ping policies
default 173
ping polling
overview 5
poll definition types 8
ping polls
changing 31
plug-in
quick reference for RCA 131
plug-in configuration parameters
setting 165
plug-in information
listing 162
plug-ins
adaptive polling 119
configuration tables 241
configuring 161
disabling 161
disco 122
enabling 161
failover 123
PostNCIMProcessing 124
SAE 125
subscriptions 163
zNetView 126
plugins 118
event map subscriptions 128
for plugins 128
RCA 130
policy throttling
administering 62
Poll Definition Editor 23
poll definition types 8
poll definitions 10
changing 36
example 43
creation 23
poll definitions *(continued)*
data labels 9
defaults 179
definition 4
deletion 49
mechanisms
ping 5
SNMP 5
overview 4
parameters 4
remote ping 6
SNMP link state 5
threshold 7
threshold expressions 189
poll policies
changing 31
defaults
basic threshold 174
generic threshold 174
ping 173
remote ping 173
reporting 177
definition 1
deletion 47
disabling 60
enabling 60
example 35
overview 1
parameters 2
refreshing 60
retrieving status 59
poll policy
scope 2
poll policy membership size
changing checking interval 64
poll policy network view updates
disabling 64
enabling 64
poll policy throttling
administering 62
poller
adding 67
removing 69
poller entity 140
configuring 169
Poller entity 133
polling
adaptive polling scenarios 51
administering network polling 59
default probes 1
managing adaptive polling 51
overview 1
polling database
embedded OQL databases 225
polling databases 213
config database 225
config.failover table 226
config.properties table 225
config.pruning table 227
config.realTimeControl table 227
NCMONITOR 213
profiling database 228
profiling.engine table 230
profiling.icmp table 229
profiling.policy table 228
profiling.snmp table 229
polling status tables 216

pollLog table 216
pollLogSummary table 218
polls
  across domains 61
  administering 59
  copying 61
  disabling 11
  enabling 11
PostNCIMProcessing plug-in 124
precedence value 131
precedence values
  defaults 132
Probe for Tivoli Netcool/OMNIbus
  configuring 195
  properties file 195
  rules file 196
process
  for event enrichment 83
processes
  events generated 208
profiling database
  for polling 228
profiling.engine table
  for polling 230
profiling.icmp table
  for polling 229
profiling.policy table
  for polling 228
profiling.snmp table
  for polling 229
programs
  get_policies.pl 61
publications vii

Q

querying
  NCIM database using OQL 158
  ObjectServer using OQL 158
queue
  outgoing on Event Gateway 91
quick reference
  event enrichment 83
  RCA plug-in 131

R

RCA 130
  and managed status 134
  checking topology paths 148
  configuring 169
  isolated suppression 144, 147
  maximum age difference for suppression 170
  precedence value 131
  precedence values 132
RCA example
  connected interfaces 142
  contained interfaces 141
  directly connected interface 146
  related logical interface 146
RCA plug-in
  configuring 169
  database tables 236
  quick reference 131
RCA stitchers 136
RCA stitchers *(continued)*
  descriptions 138
  sequence 136
reference
  event enrichment 83
  RCA plug-in 131
refreshing
  poll policies 60
remote ping policies
  default 173
remote ping poll definitions
  changing 41
  creating 29
remote ping polling
  overview 6
  restrictions 6
reporting policies
  default 177
retrieving
  status of poll policies 59
root cause analysis 141
  AMOS process 141
  config.defaults database 239
  contained interfaces 141
  downstream chassis devices 144
  mojo.events database 237
Root Cause Analysis
  isolated suppression 144, 147
root-cause analysis 130
  checking topology paths 148
  configuring 169
  maximum age difference for suppression 170
rules file processing example 197

S

SAE plug-in 125
  adding SAE types 167
  config.serviceTypes table 240
  configuring 166
  database tables 239
SAEs
  configuring summary fields 166
scenarios
  for adaptive polling 51
scope
  of poll policy 2
selection
  of event map 97
  of event map using the Event Gateway 97
selection methods
  for event map 98
sequence
  RCA stitchers 136
setting
  Event Gateway plug-in configuration parameters 165
SIGHUP command 158
SNMP
  ncmonitor database 213
SNMP link state polling
  overview 5
SNMP polling
  overview 5
  poll definition types 8
snmpAccess table 214
snmpTarget table 213
snmpUser table 215
snmpv1Sec table 215
snmpv3Sec table 215
StandardEventEnrichment.stch 113
standby filter 88
state
  of events 93
status information events 208
stitchers
  example for Event Gateway 107, 109, 111, 113
  examples of Event Gateway stitchers not used by default 114
  ExtractIfString.stch 109, 111
  for data extraction 109
  for entity retrieval 110
  for event enrichment 105, 112
  for Event Gateway 102
  for topology lookup 105
  LookupIp.stch 107
  RCA stitcher descriptions 138
  RCA stitcher sequence 136
  StandardEventEnrichment.stch 113
storage capacity
  considerations to determine increasing the storage limit 70
  example 72
  guidelines to determine increasing the storage limit 71
subscriptions
  for Event Gateway plug-ins 163
support information xi

T

tables
  expectedIps 216
  pollLog 216
  pollLogSummary 218
  snmpAccess 214
  snmpTarget 213
  snmpUser 215
  snmpv1Sec 215
  snmpv3Sec 215
threshold expression
  example 44
threshold expressions
  operators 192
  syntax 189
  use of the eval statement 189
threshold polling
  example 7
  overview 7
throttling
  administering 62
Tivoli software information center vii
Tivoli technical training x
topology lookup stitchers 105
topology pathsfor RCA 148
training, Tivoli technical x
trigger threshold
  example 187
typeface conventions xi

U

undiscoveredIps view 219
unmanagedIps view 220
unmonitoredIps view 219
unpollableIps view 224
unpolledFor15MinutesIps view 221
upstream 140
upstream devices
  root cause analysis 139

V

variables, notation for xi
views
  chassisOnlyIps 223
  delayedPollPolicies 221
  discoveredIps 222
  managementInterfaceIps 223
  undiscoveredIps 219
  unmanagedIps 220
  unmonitoredIps 219
  unpollableIps 224
  unpolledFor15MinutesIps 221
virtual routers
  entities 143
VLANs
  entities 143

Z

zNetView plug-in 126

Printed in the Republic of Ireland